(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
5 August 2004 (05.08.2004)

PCT

(10) International Publication Number
WO 2004/066278 A2

(51) International Patent Classification[7]: G11B

(21) International Application Number:
PCT/US2004/001632

(22) International Filing Date: 21 January 2004 (21.01.2004)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

60/441,810	21 January 2003 (21.01.2003)	US
10/761,884	20 January 2004 (20.01.2004)	US

(71) Applicant *(for all designated States except US)*: **EQUALLOGIC, INC.** [US/US]; 9 Townsend West, Nashua, NH 03063 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **KONING, Paul, G.** [US/US]; 408 Joe English Road, New Boston, NH 03070 (US). **HAYDEN, Peter, C.** [US/US]; 17 Purgatory Road, Mount Vernon, NH 03057 (US). **LONG, Paula** [US/US]; 25 Winchester Drive, Hollis, NH 03049 (US). **SUMAN, Daniel, E.** [US/US]; 11 Grizzley Bear Circle, Suite 201, Westford, MA 01886 (US). **LEE, Hsin, H.** [US/US]; 9 Townsend West, Nashua, NH 03063 (US).

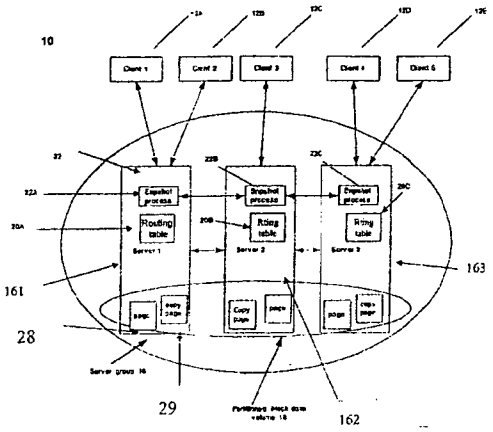
(74) Agents: **STUTIUS, Wolfgang, E.** et al.; Ropes & Gray LLP, Patent Group, One International Place, Boston, MA 02110-2624 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK,

[Continued on next page]

(54) Title: SYSTEMS FOR MANAGING DATA STORAGE



(57) Abstract: Systems for managing data storage are described. The systems manage responses to requests from a plurality of clients for access to a set of resources, and more efficiently responds to client load changes in storage area network (SAN) by migrating data blocks while providing continuous data access. The systems include a plurality of optionally equivalent servers wherein the set of resources is partitioned across these servers. Each (equivalent) server has a load monitor process that can communicate with the other load monitor processes for generating a measure of the client load on the server system and the client load on each of the respective servers. The system further comprises a resource distribution process that redistribute the client load by repartitioning the set of resources in response to the measured system load. In addition, each server may include a routing table that includes a reference for each resource that is maintained on the partitioned resource server. Requests from a client are processed as a function of the routing table to route the request to the individual server that maintains or has control over the resource of interest. For archiving purposes, a snapshot process may operate on a server, optionally in cooperation with other snapshot processes for generating state information representative of the state of the partitioned storage volume.

WO 2004/066278 A2

TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

SYSTEMS FOR MANAGING DATA STORAGE

FIELD OF THE INVENTION

The invention relates to systems and methods for managing data storage in computer networks, and more particularly to systems that store data resources across a plurality of servers and provide backup for data blocks across a plurality of servers.

BACKGROUND

The client server architecture has been one of the more successful innovations in information technology. The client server architecture allows a plurality of clients to access services and data resources maintained and/or controlled by a server. The server listens for requests from the clients and in response to the request determines whether or not the request can be satisfied, responding to the client as appropriate. A typical example of a client server system has a "file server" set up to store data files and a number of clients that can communicate with the server. Clients typically request that the server grant access to different ones of the data files maintained by the file server. If a data file is available and a client is authorized to access that data file, the server can deliver the requested data file to the server and thereby satisfy the client's request.

Although the client server architecture has worked remarkably well, it does have some drawbacks. For example, the number of clients contacting a server and the number of requests being made by individual clients can vary significantly over time. As such, a server responding to client requests may find itself inundated with a volume of requests that is impossible or nearly impossible to satisfy. To address this problem, network administrators often make sure that the server includes sufficient data processing assets to respond to anticipated peak levels of client requests. Thus, for example, the network administrator may make sure that the server comprises a sufficient number of central processing units (CPUs) with sufficient memory and storage space to handle the volume of client traffic that may arrive.

In addition, during the operation of a mass storage system to periodically gather information about how the data is stored on the system and from time-to-time to make a backup copy of the stored data. Gathering such information can be beneficial for a number of reasons, including for recovery in the event of a non-recoverable failure.

Backing up a mass storage system is typically done by reading the data stored on the mass storage system and writing it to a magnetic tape to create an archive copy of the stored data.

However, generating such archival copies can be burdensome. Many prior art backup methods require that the system be removed from ongoing (online) operations to assure the integrity and consistency of the backup copy. This is because normal backup techniques either copy the blocks from the mass storage system sequentially to a linear-access tape, or walk through the file system on the mass storage system, starting with the first block of the first file in the first directory and proceeding in order to the last block of the last file of the last directory. In either case, the backup process is unaware of updates being performed as data is being written to tape.

Thus, to permit continued, online operations while performing backup operations generates inconsistencies if the data is modified as the backup operation proceeds. Removing the storage system from continued storage operations eliminates the risk of inconsistencies arising during the system operations. However, backup operations can be time consuming therefore making removal of the system from operations undesirable.

One approach to addressing this problem, has been by creating a mirror, or identical copy, of one disk's data. When a backup operation is required, the mirror disk may be used as a static image for a storage. When the static image is no longer necessary (for example, when the tape backup has been completed), the two disks are resynchronized, by copying any changes made during the time mirroring was not active to the mirror disk, and mirroring is resumed.

Although, mirroring works well, it requires that the data stored on the system be captured accurately. Today however, new distributed storage systems are being developed that avoid the use of a centralized storage control system. These distributed systems capture the benefits of the more flexible and scalable distributed server architectures. Although very exciting, these storage systems present challenges that prior art storage systems do not. One such challenge is the ability to generate reliable and trustworthy archive copies of a data volume that has been distributed across a plurality of independently operating servers.

Note that, in this disclosure, the term "resource" is to be understood to encompass, although not be limited to the files, data blocks or pages, applications, or other services or capabilities provided by the server to clients. The term "asset" is to be understood to

encompass, although not be limited to the processing hardware, memory, storage devices, and other elements available to the server for the purpose of responding to client requests.

Even with a studied determination of needed system resources, variations in client load can still burden a server or group of servers acting in concert as a system. For example, even if sufficient hardware assets are provided in the server system, it may be the case that client requests focus on a particular file, data block within a file, or other resource maintained by the server. Thus, continuing with the above example, it is not uncommon that client requests overwhelmingly focus on a small portion of the data files maintained by the file server. Accordingly, even though the file server may have sufficient hardware assets to respond to a certain volume of client requests, if these requests are focused on a particular resource, such as a particular data file, most of the file server assets will remain idle while those assets that support the data file being targeted are over-burdened.

To address this problem, network engineers have developed load balancing systems that distribute client requests across the available assets for the purpose of distributing client demand on individual assets. To this end, the load balancing system may distribute client requests in a round-robin fashion that evenly distributes requests across the available server assets. In other practices, the network administrator sets up a replication system that can identify when a particular resource is the subject of a flurry of client requests and duplicate the targeted resource so that more of the server assets are employed in supporting client requests for that resource.

Furthermore, while servers do a good job of storing data, their assets are limited. One common technique employed today to extend server assets is to rely on peripheral storage devices such as tape libraries, RAID disks, and optical storage systems. When properly connected to servers, these storage devices are effective for backing up data online and storing large amounts of information. By connecting a number of such devices to a server, a network administrator can create a "server farm" (comprised of multiple server devices and attached storage devices) that can store a substantial amount of data. Such attached storage devices are collectively referred to as Network Attached Storage (NAS) systems.

But as server farms increase in size, and as companies rely more heavily on data-intensive applications such as multimedia, this traditional storage model is not quite as useful. This is because access to these peripheral devices can be slow, and it is not always possible for every user to easily and transparently access each storage device.

In order to address this shortfall, a number of vendors have been developing an architecture called a Storage Area Network (SAN). SANs provide more options for network storage, including much faster access to NAS-type peripheral devices. SANs further provide flexibility to create separate networks to handle large volumes of data.

A SAN is a high-speed special-purpose network or sub-network that interconnects different kinds of data storage devices with associated data servers on behalf of a larger network of users. Typically, a storage area network is part of the overall network of computing assets of an enterprise. SANs support disk mirroring, backup and restore, archiving, and retrieval of archived data, data migration from one storage device to another, and the sharing of data among different servers in a network. SANs can incorporate sub-networks containing NAS systems.

A SAN is usually clustered in close proximity to other computing resources (such as mainframes) but may also extend to remote locations for backup and archival storage, using wide area networking technologies such as asynchronous transfer mode (ATM) or Synchronous Optical Networks (SONET). A SAN can use existing communication technology such as optical fiber ESCON or Fibre Channel technology to connect storage peripherals and servers.

Although SANs hold much promise, they face a significant challenge. Bluntly, consumers expect a lot of their data storage systems. Specifically, consumers demand that SANs provide network-level scalability, service, and flexibility, while at the same time providing data access at speeds that compete with server farms.

This can be quite a challenge, particularly in multi-server environments, where a client wishing to access specific information or a specific file is redirected to a server that has the piece of the requested information or file. The client then establishes a new connection to the other server upon redirect and severs the connection to the originally contacted server. However, this approach defeats the benefit of maintaining a long-lived connection between the client and the initial server.

Another approach is "storage virtualization" or "storage partitioning" where an intermediary device is placed between the client and a set of physical (or even logical) servers, with the intermediary device providing request routing. None of the servers are aware that it is providing only a portion of the entire partitioned service, nor are any of the clients aware that the data resources are stored across multiple servers. Obviously, adding such an intermediary device adds complexity to the system.

Although the above techniques may work well in certain client server architectures, they each require additional devices or software (or both) disposed between the clients and the server assets to balance loads by coordinating client requests and data movement. As such, this central transaction point can act as a bottleneck that slows the server's response to client requests.

Furthermore, resources must be supplied continuously, in response to client requests, with strictly minimized latency. Accordingly, there is a need in the art for a method for rapidly distributing client load across a server system while at the same time providing suitable response times for incoming client resource requests and preserving a long-lived connection between the client and the initial server. There is also a need in the art for a distributed storage system that can provide reliable snapshots of the data volumes that are being maintained across the different server in the system.

SUMMARY OF THE INVENTION

The systems and methods described herein, according to one aspect of the invention, include systems for managing responses to requests from a plurality of clients for access to a set of resources. In one embodiment, the systems comprise a plurality of optionally equivalent servers wherein the set of resources is partitioned across this plurality of servers. Each equivalent server has a load monitor process that is capable of communicating with the other load monitor processes for generating a measure of the client load on the server system and the client load on each of the respective servers. The system may further comprise a resource distribution process that is responsive to the measured system load and is capable of repartitioning the set of resources to thereby redistribute the client load.

Optionally, the systems may further comprise a client distribution process that is responsive to the measured system load and is capable of repartitioning the set of client connections among the server systems to thereby redistribute the client load.

Accordingly, it will be understood that the systems and methods described herein include client distribution systems that may work with a partitioned service, wherein the partitioned service is supported by a plurality of equivalent servers each of which is responsible for a portion of the service that has been partitioned across the equivalent servers. In one embodiment each equivalent server is capable of monitoring the relative load that each of the clients that server is communicating with is placing on the system and

on that particular server. Accordingly, each equivalent server is capable of determining when a particular client would present a relative burden to service. However, for a partitioned service each client is to communicate with that equivalent server that is responsible for the resource of interest of the client. Accordingly, in one embodiment, the systems and methods described herein redistributed client load by, in part, redistributing resources across the plurality of servers.

According to another aspect of the invention, the systems and methods described herein include server systems that comprise a group of servers that support a service or resource that has been partitioned across the individual servers of the group. In one application, the systems and methods provide a partitioned storage service for providing storage services to a plurality of clients. In this embodiment, a data volume may be partitioned across a plurality of servers, with each server being responsible for a portion of the data volume. In such a partitioned storage system, the storage "volumes" may be understood as analogous to disk drives in a conventional storage system. However, in the partitioned service, the data volumes have been spread over several servers, with each server owning a portion of the data within the volume.

For the purpose of fault tolerance, data back-up, and other benefits, the partitioned storage services described herein provide a storage administrator with a snapshot process and system that creates a copy of the state of the storage volume. Typically, the snapshot process results in the creation of a second storage volume, which acts as an archive of the state of the storage system at a given time. Storage administrators may employ this archive as a recovery tool in the case that the original storage volumes fails at a later time, a backup tool for off-line backups, or for any other suitable reason.

In another embodiment, the systems and methods described herein include storage area network systems (SANs) that may be employed for providing storage assets for an enterprise. The SAN of the invention comprises a plurality of servers and/or network devices. At least a portion of the servers and network devices include a load monitor process that monitors the client load being placed on the respective server or network device. The load monitor process is further capable of communicating with other load monitor processes operating on the storage area network. Each load monitor process may be capable of generating a system-wide load analysis that indicates the client load being placed on the storage area network. Additionally, the load monitor process may be capable of generating an analysis of the client load being placed on that respective server and/or

network device. Based on the client load information observed by the load monitor process, the storage area network is capable of redistributing client load to achieve greater responsiveness to client requests. To this end, in one embodiment, the storage area network is capable of repartitioning the stored resources in order to redistribute client load. In another embodiment, the storage area network is capable of moving the client connections supported by the system for the purpose of redistributing client load across the storage area network.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other objects and advantages of the invention will be appreciated more fully from the following further description thereof, with reference to the accompanying drawings, wherein:

FIG. 1 depicts schematically the structure of a prior art system for providing access to a resource maintained on a storage area network;

FIG. 2 presents a functional block diagram of one system according to the invention;

FIG. 3 presents in more detail the system depicted in FIG. 2;

FIG. 4 is a schematic diagram of a client server architecture with servers organized in server groups;

FIG. 5 is a schematic diagram of the server groups as seen by a client;

FIG. 6 shows details of the information flow between the client and the servers of a group;

FIG. 7 is a process flow diagram for retrieving resources in a partitioned resource environment;

FIG. 8 depicts in more detail and as a functional block diagram a first embodiment of a system according to the invention;

FIG. 9 depicts an example of a routing table suitable for use with the system of FIG. 4;

FIG. 10 depicts in more detail and as a functional block diagram a second embodiment of a system according to the invention;

FIG. 11 depicts in more detail and as a functional block diagram a third embodiment of a system according to the invention;

FIG. 12 depicts one process for generating a snapshot of a storage volume supported by the system of FIG. 1; and

FIG. 13 depicts an alternate process for generating a snapshot of a storage volume.

The use of the same reference symbols in different drawings indicates similar or identical items.

DETAILED DESCRIPTION OF THE ILLUSTRATED EMBODIMENTS

To provide an overall understanding of the invention, certain illustrative embodiments will now be described. However, it will be understood by one of ordinary skill in the art that the systems and methods described herein can be adapted and modified to redistribute resources in other applications, such as distributed file systems, database applications, and/or other applications where resources are partitioned or distributed. Moreover, such other additions and modifications fall within the scope of the invention.

FIG. 1 depicts a prior art network system for supporting requests for resources from a plurality of clients 12 that are communicating across a local area network 24. Specifically, FIG. 1 depicts a plurality of clients 12, a local area network (LAN) 24, and a storage system 14 that includes an intermediary device 16 that processes requests from clients and passes them to servers 22. In one embodiment the intermediary device 16 is a switch. The system also includes a master data table 18, and a plurality of servers 22a-22n. The storage system 14 may provide a storage area network (SAN) that provide storage resources to the clients 12 operating across the LAN 24. As further shown in FIG. 1, each client 12 may make a request 20 for a resource maintained on the SAN 14. Each request 20 is delivered to the switch 16 and processed therein. During processing the clients 12 can request resources across the LAN 24 and the switch 16 employs the master data table 18 to identify which of the plurality of servers 22a through 22n has the resource being requested by the respective client 12.

In FIG. 1, the master data table 18 is depicted as a database system, however in alternative embodiments the switch 16 may employ a flat file master data table that is maintained by the switch 16. In either case, the switch 16 employs the master data table 18 to determine which of the servers 22a through 22n maintains which resources. Accordingly,

the master data table 18 acts as an index that lists the different resources maintained by the system 14 and which of the underlying servers 22a through 22n is responsible for which of the resources.

As further depicted by FIG. 1, once the switch 16 determines the appropriate server 22a through 22n for the requested resource, the retrieved resource may be passed from the identified server through the switch 16 and back to the LAN 24 for delivery (represented by arrow 21) to the appropriate client 12. Accordingly, FIG. 1 depicts that system 14 employs the switch 16 as a central gateway through which all requests from the LAN 24 are processed. The consequence of this central gateway architecture is that delivery time of resources requested by clients 12 from system 14 can be relatively long and this delivery time may increase as latency periods grow due to increased demand for resources maintained by system 14.

Turning to FIG. 2, a system 10 according to the invention is depicted. Specifically, FIG. 2 depicts a plurality of clients 12, a local area network (LAN) 24, and a server group 30 that includes plurality of servers 32A through 32N. As shown by FIG. 2, the clients 12 communicate across the LAN 24. As further shown in FIG. 2, each client 12 may make a request for a resource maintained by server group 30. In one application, the server group 30 is a storage area network (SAN) that provides network storage resources for clients 12. Accordingly, a client 12 may make a request across the (LAN) 24 that is transmitted, as depicted in FIG. 2 as request 34, to a server, such as the depicted server 32B of the SAN 30.

The depicted SAN 30 comprises a plurality of equivalent servers 32A through 32N. Each of these servers has a separate IP address and thus the system 10 appears as a storage area network that includes a plurality of different IP addresses, each of which may be employed by the clients 12 for accessing storage resources maintained by the SAN 30.

The depicted SAN 30 employs the plurality of servers 32A though 32N to partition resources across the storage area network, forming a partitioned resource set. Thus, each of the individual servers may be responsible for a portion of the resources maintained by the SAN 30. In operation, the client request 34 received by the server 32B is processed by the server 32B to determine the resource of interest to that client 12 and to determine which of the plurality of servers 32A through 32N is responsible for that particular resource. In the example depicted in FIGURES 2 and 3, the SAN 30 determines that the server 32A is responsible for the resource identified in the client request 34. As further shown by FIG. 2, the SAN 30 may optionally employ a system where, rather than have the original server 32B

respond to the client request 34, a shortcut response is employed that allows the responsible server to respond directly to the requesting client 12. Server 32A thus delivers response 38 over LAN 24 to the requesting client 12.

As discussed above, the SAN 30 depicted in FIG. 2 comprises a plurality of equivalent servers. Equivalent servers will be understood, although not limited to, server systems that expose a uniform interface to a client or clients, such as the clients 12. This is illustrated in part by FIG. 3 that presents in more detail the system depicted in FIG. 2, and shows that requests from the clients 12 may be handled by the servers, which in the depicted embodiment, return a response to the appropriate client. Each equivalent server will respond in the same manner to a request presented by any client 12, and the client 12 does not need to know which one or ones of the actual servers is handling its request and generating the response. Thus, since each server 32A through 32N presents the same response to any client 12, it is immaterial to the client 12 which of the servers 32A through 32N responds to its request.

Each of the depicted servers 32A through 32N may comprise conventional computer hardware platforms such as one of the commercially available server systems from the Sun Microsystems Inc. of Santa Clara, California. Each server executes one or more software processes for the purpose of implementing the storage area network. The SAN 30 may employ a Fibre Channel network, an arbitrated loop, or any other type of network system suitable for providing a storage area network. As further shown in FIG. 2, each server may maintain its own storage resources or may have one or more additional storage devices coupled to it. These storage devices may include, but are not limited to, RAID systems, tape library systems, disk arrays, or any other device suitable for providing storage resources to the clients 12.

It will be understood that those of ordinary skill in the art that the systems and methods of the invention are not limited to storage area network applications and may be applied to other applications where it may be more efficient for a first server to receive a request and a second server to generate and send a response to that request. Other applications may include distributed file systems, database applications, application service provider applications, or any other application that may benefit from this technique.

Referring now to FIG. 4, one or several clients 12 are connected, for example via a network 24, such as the Internet, an intranet, a WAN or LAN, or by direct connection, to servers 161, 162, and 163 that are part of a server group 116.

As described above, the depicted clients 12 can be any suitable computer system such as a PC workstation, a handheld computing device, a wireless communication device, or any other such device, equipped with a network client program capable of accessing and interacting with the server group 116 to exchange information with the server group 116.

Servers 161, 162 and 163 employed by the system 110 may be conventional, commercially available server hardware platforms, as described above. However any suitable data processing platform may be employed. Moreover, it will be understood that one or more of the servers 161, 162, or 163 may comprise a network storage device, such as a tape library, or other device, that is networked with the other servers and clients through network 24.

Each server 161, 162, and 163 may include software components for carrying out the operation and the transactions described herein, and the software architecture of the servers 161, 162, and 163 may vary according to the application. In certain embodiments, the servers 161, 162, and 163 may employ a software architecture that builds certain of the processes described below into the server's operating system, into device drivers, into application level programs, or into a software process that operates on a peripheral device (such as a tape library, RAID storage system, or another storage device or any combination thereof). In any case, it will be understood by those of ordinary skill in the art that the systems and methods described herein may be realized through many different embodiments and that the particular embodiment and practice employed will vary as a function of the application of interest. All these embodiments and practices accordingly fall within the scope of the present invention.

In operation, the clients 12 will have need of the resources partitioned across the server group 116. Accordingly, each of the clients 12 will send requests to the server group 116. The clients 12 typically act independently, and as such, the client load placed on the server group 116 will vary over time. In a typical operation, a client 12 will contact one of the servers, for example server 161, to access a resource, such as a data block, page (comprising a plurality of blocks), file, database table, application, or other resource. The contacted server 161 itself may not hold or have control over the requested resource. However, in a preferred embodiment, the server group 116 is configured to make all the

partitioned resources available to the client 12 regardless of the server that initially receives the request. For illustration, FIG. 4 shows two resources, one resource 180 that is partitioned over all three servers (servers 161, 162, 163) and another resource 170 that is partitioned over two of the three servers. In the exemplary application of the system 110 being a block data storage system, each resource 170 and 180 may represent a partitioned block data volume.

The depicted server group 116 therefore provides a block data storage service that may operate as a storage area network (SAN) comprised of a plurality of equivalent servers, servers 161, 162, and 163. Each of the servers 161, 162, and 163 may support one or more portions of the partitioned block data volumes 170 and 180. In the depicted server group 116, there are two data resources (e.g., volumes) and three servers; however there is no specific limit on the number of servers. Similarly, there is no specific limit on the number of resources or data volumes. Moreover, each resource may be contained entirely on a single server, or it may be partitioned over several servers, either all of the servers in the server group, or a subset of the server group.

In practice, there may of course be limits due to implementation considerations, for example the amount of memory assets available in the servers 161, 162 and 163 or the computational limitations of the servers 161, 162 and 163. Moreover, the grouping itself, i.e., deciding which servers will comprise a group, may in one practice involve an administrative decision. In a typical scenario, a group might at first contain only a few servers, perhaps only one. The system administrator would add servers to a group as needed to obtain the level of performance required. Increasing servers creates more space (memory, disk storage) for resources that are stored, more CPU processing capacity to act on the client requests, and more network capacity (network interfaces) to carry the requests and responses from and to the clients. It will be appreciated by those of skill in the art that the systems described herein are readily scaled to address increased client demands by adding additional servers into the group 116. However, as client load varies, the server group 116 can redistribute client load to take better advantage of the available assets in server group 116.

To this end, the server group 116, in one embodiment, comprises a plurality of equivalent servers. Each equivalent server supports a portion of the resources partitioned over the server group 116. As client requests are delivered to the equivalent servers, the equivalent servers coordinate among themselves to generate a measure of system load and to generate a measure of the client load of each of the equivalent servers. In a preferred

practice, this coordinating is transparent to the clients 12, and the servers can distribute the load among each other without causing the clients to alternate or change the way they access a resource..

Referring now to FIG. 5, a client 12 connecting to a server 161 (FIG. 4) will see the server group 116 as if the group were a single server having multiple IP addresses. The client 12 is not necessarily aware that the server group 116 is constructed out of a potentially large number of servers 161, 162, 163, nor is it aware of the partitioning of the block data volumes 170 and 180 over the several servers. A particular client 12 may have access to only a single server, through its unique IP address. As a result, the number of servers and the manner in which resources are partitioned among the servers may be changed without affecting the network environment seen by the client 12.

FIG. 6 shows the resource 180 of FIG. 5 as being partitioned across servers 161, 162 and 163. In the partitioned server group 116, any data volume may be spread over any number of servers within the server group 116. As seen in FIG. 4 and 5, one volume 170 (Resource 1) may be spread over servers 162, 163, whereas another volume 180 (Resource 2) may be spread over servers 161, 162, 163. Advantageously, the respective volumes may be arranged in fixed-size groups of blocks, also referred to as "pages," wherein an exemplary page contains 8192 blocks. Other suitable page sizes may be employed, and pages comprising variable numbers of blocks (rather than fixed) are also possible.

In an exemplary embodiment, each server in the group 116 contains a routing table 165 for each volume, with the routing table 165 identifying the server on which a specific page of a specific volume can be found. For example, when the server 161 receives a request from a client 12 for volume 3, block 93847, the server 161 calculates the page number (page 11 in this example for the page size of 8192) and looks up in the routing table 165 the location or number of the server that contains page 11. If server 163 contains page 11, the request is forwarded to server 163, which reads the data and returns the data to the server 161. Server 161 then sends the requested data to the client 12. The response may be returned to the client 12 via the same server 161 that received the request from the client 12. Alternatively, the short-cut approach described above may be used.

Accordingly, it is immaterial to the client 12 as to which server 161, 162, 163 has the resource of interest to the client 12. As described above, the servers 162, 162 and 163 will employ the routing tables to service the client request, and the client 12 need not know ahead of time which server is associated with the requested resource. This allows portions

of the resource to exist at different servers. It also allows resources, or portions thereof, to be moved while the client 12 is connected to the partitioned server group 116. This latter type of resource re-partitioning is referred to herein as "block data migration" in the case of moving parts of resources consisting of data blocks or pages. One of ordinary skill in the art will of course see that resource parts consisting of other types of resources (discussed elsewhere in this disclosure) may also be moved by similar means. Accordingly, the invention is not limited to any particular type of resource.

Data may be moved upon command of an administrator or automatically by storage load balancing mechanisms such as those discussed herein. Typically, such movement or migration of data resources is done in groups of blocks referred to as pages.

When a page is moved from one equivalent server to another, it is important for all of the data, including that in the page being moved, to be continuously accessible to the clients so as not to introduce or increase response time latency. In the case of manual moves, as implemented in some servers seen today, the manual migration interrupts service to the clients. As this is generally considered unacceptable, automatic moves that do not result in service interruptions are preferable. In such automatic migrations, the movement must needs be transparent to the clients.

According to one embodiment of the present invention, a page to be migrated is considered initially "owned" by the originating server (i.e., the server on which the data is initially stored) while the move is in progress. Routing of client read requests continue to go through this originating server.

Requests to write new data into the target page are handled specially: data is written to both the page location at the originating server and to the new (copy) page location at the destination server. In this way, a consistent image of the page will end up at the destination server even if multiple write requests are processed during the move. In one embodiment, it is the resource transfer process 240 depicted in FIG. 8 that carries out this operation. A more elaborate approach may be used when pages become large. In such cases, the migration may be done in pieces: a write to a piece that has already been moved is simply redirected to the destination server; a write to a piece currently being moved goes to both servers as before. Obviously, a write to a piece not yet moved may be processed by the originating server.

Such write processing approaches are necessary to support the actions required if a failure should occur during the move, such as a power outage. If the page is moved as a

single unit, an aborted (failed) write can begin over again from the beginning. If the page is moved in pieces, the move can be restarted from the piece that was in transit at the failure. It is the possibility of restart that makes it necessary to write data to both the originating and destination servers.

Table 1 shows the sequence of block data migration stages for a unit block data move from a server A to Server B; Table 2 shows the same information for a piece-wise block data move.

Table 1

Stage	Restart Stage	Destination	Action	Read	Write
1	N/A	Server A	Not started	Server A	Server A
2	2	Server A	Start move	Server A	Servers A and B
3	2	Server A	Finish Move	Server A	Servers A and B
4	N/A	Server B	Routing Table updated	Server B	Server B

Table 2

Stage	Restart Stage	Destination	Action	Read	Write
1	N/A	Server A	Not started	Server A	Server A
2	2		Start move, piece 1	Server A	Servers A and B for piece 1, server A for others
3	2	Server A	Finish move, piece 1	Server B for piece 1, server A for others	Server B for piece 1, server A for others
4	4		Start move, piece 2	Server B for piece 1, server A for others	Server B for piece 1, servers A and B for piece 2, server A for others
5	4		Finish move, piece 2	Server B for pieces 1 and 2, server A for others	Server B for pieces 1 and 2, server A for others
...	(repeat as needed for all pieces)				
n	N/A	Server B	Routing Table updated	Server B	Server B

Upon moving a resource, the routing tables 165 (referring again to FIG. 9) are updated as necessary (through means well-known in the art) and subsequent client requests will be forwarded to the server now responsible for handling that request. Note that, at least among servers containing the same resource 170 or 180, the routing tables 165 may be identical subject to propagation delays.

In some embodiments, once the routing tables are updated, the page at the originating server (or "source" resource) is deleted by standard means. Additionally, a flag or other marker is set in the originating server for the originating page location to denote, at least temporarily, that that data is no longer valid. Any latent read or write requests still destined for the originating server will thus trigger an error and subsequent retry, rather than reading the expired data on that server. By the time any such retries return, they will encounter the updated routing tables and be appropriately directed to the destination server. In no case are duplicated, replicated, or shadowed copies of the block data (as those terms are known in the art) left on in the server group. Optionally, in other embodiments the originating server may retain a pointer or other indicator to the destination server. The originating server may, for some selected period of time, forward requests, including but not being limited to, read and write requests, to the destination server. In this optional embodiment, the client 12 does not receive an error when the requests are latest or arrive at the originating server because some of the routing tables in the group have not yet been updated. Requests can be handled at both the originating server and the destination server. This lazy updating process eliminates or reduces the need to synchronize the processing of client requests with routing table updates. The routing table updates occur in the background.

FIG. 7 depicts an exemplary request handling process 400 for handling client requests in a partitioned server environment. The request handling process 400 begins at 410 by receiving a request for a resource, such as a file or blocks of a file, at 420. The request handling process 400 examines the routing table, in operation 430, to determine at which server the requested resource is located. If the requested resource is present at the initial server, the initial server returns the requested resource to the client 12, at 480, and the process 400 terminates at 490. Conversely, if the requested resource is not present at the initial server, the server will use the data from the routing table to determine which server actually holds the resource requested by the client, operation 450. The request is then forwarded to the server that holds the requested resource, operation 460, which returns the requested resource to the initial server, operation 480. The process 400 then goes to 480 as before, to have the initial server forward the requested resource to the client 12, and the process 400 terminates, at 490.

Accordingly, one of ordinary skill in the act will see that the system and methods described herein are capable of migrating one or more partitioned resources over a plurality

of servers, thereby providing a server group capable of handling requests from multiple clients. The resources so migrated over the several servers can be directories, individual files within a directory, blocks within a file or any combination thereof. Other partitioned services may be realized. For example, it may be possible to partition a database in an analogous fashion or to provide a distributed file system, or a distributed or partitioned server that supports applications being delivered over the Internet. In general, the approach can be applied to any service where a client request can be interpreted as a request for a piece of the total resource.

Turning now to FIG. 8, one particular embodiment of the system 500 is depicted wherein the system is capable of redistributing client load to provide more efficient service. Specifically, FIG. 8 depicts a system 500 wherein the clients 12A through 12E communicate with the server block 116. The server block 116 includes three equivalent servers, server 161, 162, and 163, in that each of the servers will provide substantially the same response to the same request from a client. Typically, it will produce the identical response, subject to differences arising due to propagation delay or response timing. As such, from the perspective of the clients 12, the server group 116 appears to be a single server system that provides multiple network or IP addresses for communicating with clients 12A-12E.

Each server includes a routing table, depicted as routing tables 200A, 200B and 200C, a load monitor process 220A, 220B and 220C respectively, a client allocation process 320A, 320B, and 320C, a client distribution process 300A, 300B and 300C and a resource transfer process, 240A, 240B and 240C respectively. Further, and for the purpose of illustration only, FIG. 8 represents the resources as pages of data 280 that may be transferred from one server to another server.

As shown by arrows in FIG. 8, each of the routing tables 200A, 200B, and 200C are capable of communicating with each other for the purpose of sharing information. As described above, the routing tables may track which of the individual equivalent servers is responsible for a particular resource maintained by the server group 116. Because each of the equivalent servers 161, 162 and 163 are capable of providing the same response to the same request from a client 12, routing tables 200A, 200B, and 200C (respectively) coordinate with each other to provide a global database of the different resources and the specific equivalent servers that are responsible for those resources.

FIG. 9 depicts one example of a routing table 200A and the information stored therein. As depicted in FIG. 9, each routing table includes an identifier for each of the

equivalent servers 161, 162 and 163 that support the partitioned data block storage group 116. Additionally, each of the routing tables includes a table that identifies those data blocks associated with each of the respective equivalent servers. In the routing table embodiment depicted by FIG. 9, the equivalent servers support two partitioned volumes. A first one of the volumes is distributed or partitioned across all three equivalent servers 161, 162, and 163. The second partitioned volume is partitioned across two of the equivalent servers, servers 162 and 163 respectively.

In operation, each of the depicted servers 161, 162, and 163 is capable of monitoring the complete load that is placed on the server group 116 as well as the load from each client and the individual client load that is being handled by each of the respective servers 161, 162 and 163. To this end, each of the servers 161, 162 and 163 include a load monitoring process 220A, 220B, and 220C respectively. As described above, the load monitor processes 220A, 220B and 220C are capable of communicating among each other. This is illustrated in FIG. 8 by the bidirectional lines that couple the load monitor processes on the different servers 161, 162 and 163.

Each of the depicted load monitor processes may be software processes executing on the respective servers and monitoring the client requests being handled by the respective server. The load monitors may monitor the number of individual clients 12 being handled by the respective server, the number of requests being handled by each and all of the clients 12, and/or other information such as the data access patterns (predominantly sequential data access, predominantly random data access, or neither).

Accordingly, the load monitor process 220A is capable of generating information representative of the client load applied to the server 161 and is capable of corresponding with the load monitor 220B of server 162. In turn, the load monitor process 220B of server 162 may communicate with the load monitor process 220C of server 163, and load monitor process 220C may communicate with process 220A (not shown). By allowing for communication between the different load monitor processes 220A, 220B, and 220C, the load monitor processes may determine the system-wide load applied to the server group 116 by the clients 12.

In this example, the client 12C may be continually requesting access to the same resource. For example, such a resource may be the page 280, maintained by the server 161. That load in addition to all the other requests may be such that server 161 is carrying an excessive fraction of the total system traffic, while server 162 is carrying less than the

expected fraction. Therefore the load monitoring and resource allocation processes conclude that the page 280 should be moved to server 162, and the client distribution process 300A can activate the block data migration process 350 (described above) that transfers page 280 from server 161 to server 162. Accordingly, in the embodiment depicted in FIG. 8 the client distribution process 300A cooperates with the resource transfer process 240A to re-partition the resources in a manner that is more likely to cause client 12C to continually make requests to server 162 as opposed to server 161.

Once the resource 280 has been transferred to server 162, the routing table 200B can update itself (by standard means well-known in the art) and update the routing tables 200A and 200C accordingly, again by standard means well-known in the art. In this way, the resources may be repartitioned across the servers 161, 162 and 163 in a manner that redistribute client load as well.

Referring now back to FIG. 4, the systems and methods can also be used for providing a more efficient operation of a partitioned service.

In this embodiment, the server group 16 provides a block data storage service that may operate as a storage area network (SAN) comprised of a plurality of equivalent servers, servers 161, 162 and 163. Each of the servers 161, 162 and 163 may support one or more portions of the partitioned block data volumes 188 and 170. In the depicted system 110, there are two data volumes and three servers, however there is no specific limit on the number of servers. Similarly, there is no specific limit on the number of resources or data volumes. Moreover, each data volume may be contained entirely on a single server, or it may be partitioned over several servers, either all of the servers in the server group, or a subset of the server group. In practice, there may of course be limits due to implementation considerations, for example the amount of memory available in the servers 161, 162 and 163 or the computational limitations of the servers 161, 162 and 163. Moreover, the grouping itself, i.e., deciding which servers will comprise a group, may in one practice comprise an administrative decision. In a typical scenario, a group might at first contain only a few servers, perhaps only one. The system administrator would add servers to a group as needed to obtain the level of service required. Increasing servers creates more space (memory, disk storage) for resources that are stored, more CPU processing capacity to act on the client requests, and more network capacity (network interfaces) to carry the requests and responses from and to the clients. It will be appreciated by those of skill in the art that the systems described herein are readily scaled to address increased client demands by adding additional

servers into the group 116. However, as client load varies, the system 110 as described below can redistribute client load to take better advantage of the available resources in server group 116. To this end, the system 110 in one embodiment, comprises a plurality of equivalent servers. Each equivalent server supports a portion of the resources partitioned over the server group 116. As client requests are delivered to the equivalent servers, the equivalent servers coordinate among themselves to generate a measure of system load and to generate a measure of the client load of each of the equivalent servers. In a preferred practice, this coordinating is done in a manner that is transparent to the clients 12, so that the clients 12 see only requests and responses traveling between the clients 12 and server group 16.

Referring now back to FIG. 5, a client 12 connecting to a server 161 (FIG. 4) will see the server group 116 as if the group were a single server having multiple IP addresses. The client 12 is not aware that the server group 116 is constructed out of a potentially large number of servers 161, 162, 163, nor is it aware of the partitioning of the block data volumes 170, 180 over the several servers 161, 162, 163. As a result, the number of servers and the manner in which resources are partitioned among the servers may be changed without affecting the network environment seen by the client 12.

Referring now to FIG. 6, in the partitioned server group 116, any volume may be spread over any number of servers within the group 116. As seen in FIGS. 4 and 5, one volume 170 (Resource 1) may be spread over servers 162, 163, whereas another volume 180 (Resource 2) may be spread over servers 161, 162, 163. Advantageously, the respective volumes are arranged in fixed-size groups of blocks, also referred to as "pages", wherein an exemplary page contains 8192 blocks. Other suitable page sizes may be employed. In an exemplary embodiment, each server in the group 116 contains a routing table 165 for each volume, with the routing table 165 identifying the server on which a specific page of a specific volume can be found. For example, when the server 161 receives a request from a client 12 for volume 3, block 93847, the server 161 calculates the page number (page 11 in this example for the page size of 8192) and looks up in the routing table 165 the location or number of the server that contains page 11. If server 163 contains page 11, the request is forwarded to server 163, which reads the data and returns the data to the server 161. Server 161 then send the requested data to the client 12. In other words, the response is always returned to the client 12 via the same server 161 that received the request from the client 12.

It is transparent to the client 12 to which server 161, 162, 163 he is connected. Instead,

the client only sees the servers in the server group 116 and requests the resources of the server group 116. It should be noted here that the routing of client requests is done separately for each request. This allows portions of the resource to exist at different servers. It also allows resources, or portions thereof, to be moved while the client is connected to the server group 116 – if that is done, the routing tables 165 are updated as necessary and subsequent client requests will be forwarded to the server now responsible for handling that request. At least within a resource 170 or 180, the routing tables 165 are identical. The described invention is different from a "redirect" mechanism, wherein a server determines that it is unable to handle requests from a client, and redirects the client to the server that can do so. The client then establishes a new connection to another server. Since establishing a connection is relatively inefficient, the redirect mechanism is ill suited for handling frequent requests.

FIG. 7 depicts an exemplary request handling process 400 for handling client requests in a partitioned server environment. The request handling process 400 begins at 410 by receiving a request for a resource, such as a file or blocks of a file, at 420. The request handling process 400 checks, in operation 430, if the requested resource is present at the initial server that received the request from the client 12 examines the routing table, in operation 430, to determine at which server the requested resource is located. If the requested resource is present at the initial server, the initial server returns the requested resource to the client 12, at 480, and the process 400 terminates at 490. Conversely, if the requested resource is not present at the initial server, the server will consult a routing table, operation 440,use the data from the routing table to determine which server actually holds the resource requested by the client, operation 450. The request is then forwarded to the server that holds the requested resource, operation 460, which returns the requested resource to the initial server, operation 480. The process 400 then goes to 480 as before, to have the initial server forward the requested resource to the client 12, and the process 400 terminates, at 490.

The resources spread over the several servers can be directories, individual files within a directory, or even blocks within a file. Other partitioned services could be contemplated. For example, it may be possible to partition a database in an analogous fashion or to provide a distributed file system, or a distributed or partitioned server that supports applications being delivered over the Internet. In general, the approach can be applied to any service where a client request can be interpreted as a request for a piece of the total resource, and

operations on the pieces do not require global coordination among all the pieces.

Turning now to FIG. 10, one particular embodiment of a block data service system 10 is depicted. Specifically, FIG. 10 depicts the system 10 wherein the client 12 communicates with the server group 16. The server group 16 includes three servers, server 161, 162 and 163. Each server includes a routing table depicted as routing tables 20A, 20B and 20C. In addition to the routing tables, each of the equivalent servers 161, 162 and 163 are shown in FIG. 10 as including a load monitor process, 22A, 22B and 22C respectively.

As shown in FIG. 10, each of the equivalent servers 161, 162 and 163 may include a routing table 20A, 20B and 20C respectively. As shown in FIG. 10, each of the routing tables 20A, 20B and 20C are capable of communicating with each other for the purposes of sharing information. As described above, the routing tables can track which of the individual equivalent servers is responsible for a particular resource maintained by the server group 16. In the embodiment shown in FIG. 10 the server group 16 may be a SAN, or part of a SAN, wherein each of the equivalent servers 161, 162 and 163 has an individual IP address that may be employed by a client 12 for accessing that particular equivalent server on the SAN. As further described above, each of the equivalent servers 161, 162 and 163 is capable of providing the same response to the same request from a client 12. To that end, the routing tables of the individual equivalent 161, 162 and 163 coordinate with each other to provide a global database of the different resources, and this exemplary embodiments data blocks, pages or other organizations of data blocks, and the individual equivalent servers that are responsible for those respective data blocks, pages, files or other storage organization.

Returning now to FIG. 9, there is depicted an exemplary routing table. Each routing table in the server group 16, such as table 20A, includes an identifier (Server ID) for each of the equivalent servers 161, 162 and 163 that support the partitioned data block storage service. Additionally, each of the routing tables includes a table that identifies those data blocks pages associated with each of the respective equivalent servers. In the embodiment depicted by FIG. 9, the equivalent servers support two partitioned volumes. A first one of the volumes, Volume 18, is distributed or partitioned across all three equivalent servers 161, 162 and 163. The second partitioned volume, Volume 17, is partitioned across two of the equivalent servers, servers 162 and 163 respectively.

The routing tables may be employed by the system 10 to balance client load across the available servers.

The load monitor processes 22A, 22B and 22C each observe the request patterns arriving at their respective equivalent servers to determine to determine whether patterns or requests from clients 12 are being forwarded to the SAN and whether these patterns can be served more efficiently or reliably by a different arrangement of client connections to the several servers. In one embodiment, the load monitor processes 22A, 22B and 22C monitor client requests coming to their respective equivalent servers. In one embodiment, the load monitor processes each build a table representative of the different requests that have been seen by the individual request monitor processes. Each of the load monitor processes 22A, 22B and 22C are capable of communicating between themselves for the purpose of building a global database of requests seen by each of the equivalent servers. Accordingly, in this embodiment each of the load monitor processes is capable of integrating request data from each of the equivalent servers 161, 162 and 163 in generating a global request database representative of the request traffic seen by the entire block data storage system 16. In one embodiment, this global request database is made available to the client distribution processes 30A, 30B and 30C for their use in determining whether a more efficient or reliable arrangement of client connections is available.

FIG. 10 illustrates pictorially that the server group 16 may be capable of redistributing client load by having client 12C, which was originally communicating with server 161, redistributed to server 162. To this end, FIG. 10 depicts an initial condition wherein the server 161 is communicating with clients 12A, 12B, and 12C. This is depicted by the bidirectional arrows coupling the server 161 to the respective clients 12A, 12B, and 12C. As further shown in FIG. 10, in an initial condition, clients 12D and 12E are communicating with server 163 and no client (during the initial condition) is communicating with server 162. Accordingly, during this initial condition, server 161 is supporting requests from three clients, clients 12A, 12B, and 12C. Server 162 is not servicing or responding to requests from any of the clients.

Accordingly, in this initial condition the server group 16 may determine that server 161 is overly burdened or asset constrained. This determination may result from an analysis that server 161 is overly utilized given the assets it has available. For example, it could be that the server 161 has limited memory and that the requests being generated by clients 12A, 12B, and 12C have overburdened the memory assets available to server 161. Thus, server 161 may be responding to client requests at a level of performance that is below an acceptable limit. Alternatively, it may be determined that server 161, although performing

and responding to client requests at an acceptable level, is overly burdened with respect to the client load (or bandwidth) being carried by server 162. Accordingly, the client distribution process 30 of the server group 16 may make a determination that overall efficiency may be improved by redistributing client load from its initial condition to one wherein server 162 services requests from client 12C. Considerations that drive the load balancing decision may vary and some examples are the desire to reduce routing: for example if one server is the destination of a significantly larger fraction of requests than the others on which portions of the resource (e.g., volume) resides, it may be advantageous to move the connection to that server. Or to further have balancing of server communications load: if the total communications load on a server is substantially greater than that on some other, it may be useful to move some of the connections from the highly loaded server to the lightly loaded one, and balancing of resource access load (e.g., disk I/O load) -- as preceding but for disk I/O load rather than comm load. This is an optimization process that involves multiple dimensions, and the specific decisions made for a given set of measurements may depend on administrative policies, historical data about client activity, the capabilities of the various servers and network components, etc.

To this end, FIG. 10 depicts this redistribution of client load by illustrating a connection 325 (depicted by a dotted bi-directional arrow) between client 12C and server 162. It will be understood that after redistribution of the client load, the communication path between the client 12C and server 161 may terminate.

Balancing of client load is also applicable to new connections from new clients. When a client 12F determines that it needs to access the resources provided by server group 16, it establishes an initial connection to that group. This connection will terminate at one of the servers 161, 162, or 163. Since the group appears as a single system to the client, it will not be aware of the distinction between the addresses for 161, 162, and 163, and therefore the choice of connection endpoint may be random, round robin, or fixed, but will not be responsive to the current load patterns among the servers in group 16.

When this initial client connection is received, the receiving server can at that time make a client load balancing decision. If this is done, the result may be that a more appropriate server is chosen to terminate the new connection, and the client connection is moved accordingly. The load balancing decision in this case may be based on the general level of loading at the various servers, the specific category of resource requested by the client 12F when it established the connection, historic data available to the load monitors in

the server group 16 relating to previous access patterns from server 12F, policy parameters established by the administrator of server group 16, etc.

Another consideration in handling initial client connections is the distribution of the requested resource. As stated earlier, a given resource may be distributed over a proper subset of the server group. If so, it may happen that the server initially picked by client 12F for its connection serves no part of the requested resource. While it is possible to accept such a connection, it is not a particularly efficient arrangement because in that case all requests from the client, not merely a fraction of them, will require forwarding. For this reason it is useful to choose the server for the initial client connection only from among the subset of servers in server group 16 that actually serve at least some portion of the resource requested by new client 12F.

This decision can be made efficiently by the introduction of a second routing database. The routing database described earlier specifies the precise location of each separately moveable portion of the resource of interest. Copies of that routing database need to be available at each server that terminates a client connection on which that client is requesting access to the resource in question. The connection balancing routing database simply states for a given resource as a whole which servers among those in server group 16 currently provide some portion of that resource. For example, the connection balancing routing database to describe the resource arrangement shown in Figure 1 consists of two entries: the one for resource 17 lists servers 162 and 163, and the one for resource 18 lists servers 161, 162, and 163.

Referring now back to FIGS. 4 to 7, one of ordinary skill in the act will see that the systems and methods can also be used for the system and methods described herein are capable of partitioning one or more resources over a plurality of servers thereby providing a server group capable of handling requests from multiple clients. Additionally, the above description illustrates that the systems and methods described herein can redistribute or repartition the resource to change how portions of the resource are distributed or spread across the server group. The resources spread over the several servers can be directories, individual files within a directory, blocks within a file or any combination thereof. Other partitioned services may be realized. For example, it may be possible to partition a database in an analogous fashion or to provide a distributed file system, or a distributed or partitioned server that supports applications being delivered over the Internet. In general, the approach

may be applied to any service where a client request can be interpreted as a request for a piece of the total resource.

Turning now to FIG. 11, one particular embodiment of the system 10 is depicted wherein the system is capable of generating a distributed snapshot of the storage volume 18 partitioned across the servers 161, 162 and 163. Specifically, FIG. 11 depicts the system 10 wherein the clients 12 communicate with the server group 16. The server group 16 includes three servers, server 161, 162 and 163. In the embodiment of FIG. 11 the servers 161, 162 and 163 are equivalent servers, in that each of the servers will provide substantially the same resource to the same request from a client. As such, from the perspective of the clients 12, the server group 16 appears to be a single server system that provides multiple network or IP addresses for communicating with clients 12. Each server includes a routing table, depicted as routing tables 20A, 20B and 20C, and a snapshot process 22A, 22B and 22C respectively. Further, and for the purpose of illustration only, the FIG. 11 represents the resources as pages of data 28 that may be copied to generate a second storage volume that is an image of the original storage volume 18.

As shown in FIG. 11, each of the routing tables 20A, 20B and 20C are capable of communicating with each other for the purpose of sharing information. As described above, the routing tables may track which of the individual equivalent servers is responsible for a particular resource maintained by the server group 16. In the embodiment shown in FIG. 11 the server group 16 may form a SAN wherein each of the equivalent servers 161, 162 and 163 has an individual IP address that may be employed by a client 12 for accessing that particular equivalent server on the SAN. As further described above, each of the equivalent servers 161, 162 and 163 may be capable of providing the same response to the same request from a client 12. To that end, the routing tables 20A, 20B and 20C of the individual equivalent 161, 162 and 163 coordinate with each other to provide a global database of the different resources, and the specific equivalent servers that are responsible for those resources.

As depicted in FIG. 9, each routing table includes an identifier (Server ID) for each of the equivalent servers 161, 162 and 163 that support the partitioned data block storage service. Additionally, each of the routing tables includes a table that identifies those data pages associated with each of the respective equivalent servers. As depicted by FIG. 9, the equivalent servers support two partitioned volumes. A first one of the volumes, Volume 18,

is distributed or partitioned across all three equivalent servers 161, 162 and 163. The second partitioned volume, Volume 17, is partitioned across two of the equivalent servers, servers 162 and 163 respectively.

Returning now again to FIG. 11, it can be seen that each of the equivalent servers 161, 162 and 163 includes a snapshot process 22a, 22b and 22c, respectively. Each snapshot process may be a computer process operating on the server system and designed for generating a snapshot of that portion of that storage volume which is maintained by its respective server. Accordingly, the snapshot process 22a depicted in Figure 5 may be responsible for generating a copy of that portion of storage volume 18 that is maintained by server 161. This operation is depicted, at least in part, by FIG. 11 showing a page 28 and a copy of the page 29.

In operation, each of the equivalent servers 161, 162 and 163 is generally capable of acting independently. Accordingly, the snapshot processes 22a, 22b and 22c must act in a coordinated manner to create an accurate snapshot of the storage volume 18 at a particular point in time. This need for coordination arises, at least in part, from the fact that write requests may be issued from the client's 12a through 12e at any time and to any of the servers 161, 162 and 163. Accordingly, write requests will be received by individual ones of the servers 161, 162 and 163 during the time that a snapshot process has begun. To prevent a snapshot process from generating unacceptable or unexpected results, the snapshot processes 22a, 22b and 22c coordinate their operation with each other for the purposes of generating state information that is representative of the state of the partitioned storage volume 18 at a particular point in time. Specifically, in one practice a time parameter is selected such that there is a time "T", shortly after the issuing of the command to create a snapshot, such that all write operations for which completion is indicated to the client 12 prior to "T" are included in the snapshot, and no write operations for which completion is indicated after "T" are not included in the snapshot.

To this end, each snapshot process 22a, 22b and 22c is capable of receiving a request from an administrator to create a snapshot of the storage volume 18. The snapshot process includes a coordinating process that will generate commands for coordinating the activities and operation of the snapshot processes operating on other servers that are supporting the storage volume of interest to the administrator. In the example depicted in FIG. 11, an administrator may issue a snapshot command to the snapshot process 22b operating on server 162. The snapshot command may request the snapshot process 22b to create a

snapshot of the storage volume 18. The snapshot process 22b can access the routing table 22b to determine those servers in the server group 16 that are supporting at least a portion of the data blocks within storage volume 18. The snapshot process 22b may then issue a command to each of the servers supporting a portion of the storage volume 18. In the example of FIG. 11, each of the servers 161, 162 and 163 are supporting a portion of the storage volume 18. Accordingly, the snapshot process 22b may issue a command to each of the snapshot processes 22a and 22b to prepare for creating a snapshot. At the same time, the snapshot process 22b can begin itself to prepare to create a snapshot of that portion of the storage volume 18 maintained on server 162.

In one practice, shown in Figure 7, in response to receiving the command from snapshot process 22b to prepare for creating a snapshot, each of the snapshot processes 22a, 22b and 22 c, may suspend all requests received by clients impending execution. This may include write requests and read requests as well as any other requests appropriate for the application. To this end, each snapshot process 22a, 22b and 22c may include a request control process that allows the snapshot process to process requests being carried out by its respective server and suspend operation of those requests, thereby putting a hold on write operations that may change the state of the storage volume 18.

Once the snapshot process has suspended processing of requests, it may generate a reply to the coordinating snapshot process 22b indicating that the server is ready to begin taking a snapshot of the storage volume 18. Once the coordinating snapshot process 22b has received a ready signal from each of the servers 22a and 22c and has determined that it is also ready for a snapshot operation, the coordinating snapshot process 22b may issue a snapshot command to each of the appropriate servers. In response to receiving the snapshot commands, the server may activate, optionally, an archive process that generates state information that is representative of a copy of the data blocks of volume 18 maintained by that respective server In one practice and one embodiment, a mirror image is created, through a "copy on write" process such that the portions (pages) of the volume which have not changed since the creation of the snapshot are recorded once. That mirror image may be transferred to tape or other archival storage at a later time if desired. Such techniques are known in the art, and the technique employed may vary according to the application and as appropriate given the volume of the mirror image and other similar criteria.

Once the state information has been created, the snapshot process is terminated and the servers may release any suspended or pending requests for processing.

FIG. 12 depicts one process according to the invention for generating a snapshot image of a data volume that has been partitioned across the servers 161, 162 and 163. As described more fully herein the distributed snapshot 70 depicted by FIG. 12 allows a storage administrator to generate information representative of the state of the storage volume 18 at a particular point and time. The state information generated may include information such as the file structure, meta-data about the stored data, copies of the data maintained by the partitioned storage volume or copies of portions of the storage volume, or other such information. Accordingly, it will be understood that the snapshot process described herein has many applications including applications wherein information is generated about the structure of the partitioned data volume and stored for later use as well as applications wherein a complete archived copy of the partitioned storage volume is created. The distributed snapshot process described herein may be employed in other applications and such other applications shall be understood to fall within the scope of the invention.

FIG. 12 depicts a time/space diagram that shows a sequence of operations that implement a snapshot request for the purpose of generating state information of a partitioned storage volume or storage volumes. In particular, FIG. 12 depicts a multistage process 70 that creates a consistent distributed snapshot of the storage volume. To this end, FIG. 12 depicts three vertical lines to represent the three servers, 162, 162 and 163 shown in Figure 5. Arrows 72 through 78 depict write requests issued from one or more clients 12, and arrows 82-88 represent responses from respective ones of the servers 161, 162 and 163.

As shown in FIG. 12, the process 70 begins when a snapshot command is issued from an administrator. In this case, the snapshot command is issued from the administrator and delivered to server 162. The snapshot command is depicted as arrow 90 directed to server 162. As shown in FIG. 12, the snapshot process executing on server 162 responds to the snapshot command by generating commands for coordinating the operation of the other servers 161 and 163. The commands will coordinate the snapshot processes executed on servers 161 and 163 and generate state information representative of the state of the data maintained by each of the respective servers as part of the storage volume 18.

As further shown in FIG. 12, the snapshot process executing on server 162 issues a prepare command 92 and 94 to each of the respective servers 161 and 163. The snapshot

processes operating on each of these respective servers 161 and 163 respond to the prepare command by holding pending requests received from clients prior to the arrival of the "prepare" command (e.g., request 78) and requests received subsequent to the "prepare" command (e.g., request 76).

Once requests have been held, the servers 161 and 163 reply to the server 162 that issued the prepare command indicating that the respective servers 161 and 163 have suspended all pending requests. The server 162 acting as the coordinating server then issues the snapshot command to each of the servers. This is shown in FIG. 12 by the arrows 98 and 100.

In response to the snapshot command, servers 161 and 163, as well as server 162, create a snapshot of the portion of the data volume maintained by that respective server. The snapshot information may then be stored in a data file on each of the respective servers. In an optional practice, the snapshot processes on each of the servers 161, 162, and 163, may generate an archive copy of the data volume. The archive copy may be transferred to a tape storage device, or some other mass storage device.

The snapshot generated will contain all of the request completed in the region 104 and none of those completed in region 110.

FIG. 13 depicts an alternative embodiment of a process for generating a snapshot of a storage volume. Specifically, FIG. 13 depicts a space-time diagram that shows a process 120 as it occurs over three time periods. These time periods are depicted in FIG. 13 as different shaded regions within the space-time diagram and are labeled as time periods 122, 124 and 126. Time period 122 occurs before the time at which an administrator issues a snapshot request, time period 124 occurs between the time period that the snapshot request is issued and the snapshot operation begins, and time period 128 occurs after the snapshot has been created. The request for a snapshot operation is shown by the arrow 140 and different write requests are illustrated by the arrows 130 through 138. Responses to the write requests are illustrated by arrows 131, 133, 135, 137 and 139. As in FIG. 12, the three servers of the system 10 depicted FIG. 4 are shown by the vertical lines which are labeled server 161, 162 and 163 respectively.

The process 120 depicted by FIG. 13 illustrates the creation of a consistent distributed snapshot through the use of time stamps and synchronized system clocks. More

particularly, the process 120 illustrates that the servers 161, 162 and 163 can receive a plurality of write requests, each of which can arrive at one of the respective servers at any particular time. This is shown in FIG. 13 by the write requests 130, 132 and 136 which occur during the time period 122. As further shown in FIG. 13 write request 134 may arrive during the time period 124 and write request 138 may arrive during the time period 128. Accordingly, the process 120 depicted in FIG. 13 is designed to handle write requests that can occur before, during and after the snapshot process.

The snapshot process begins when a snapshot request 140 is received by at least one of the servers 161, 162 and 163. FIG. 13 depicts snapshot request 140 being sent from an administrator to the server 162. Upon receipt of the snapshot request 140, the snapshot process operating on the server 162 may issue "prepare" commands to the other servers that are supporting the data volume for which the snapshot is being created. The prepare command is depicted by the arrows 142 which is sent from server 162 to the servers 161 and 163. Upon receipt of the prepare command, the servers 161 and 163 as well as server 162, prepare for a snapshot. In this case, requests that are still pending at the servers are allowed to proceed and can be acknowledged as soon as they finish, as it is not necessary to hold them pending. Instead the servers 161, 162 and 163 determine the time at which each such request was processed and time stamps each of the respective requests. In the example depicted by FIG. 13, this time stamping is done to write requests 136, 134 and 138, all of which are pending or received after the snapshot request 140 has been received by server 162. Once the coordinating server 162 receives a "ready" response from each of the servers 161 and 163, the coordinating server 162 generates a command to take a snapshot and transmits this command to the waiting servers 161 and 163. This command includes a time-stamp, which is the current time. This is illustrated in FIG. 13 by the arrows 160 and 162 that represent commands to the servers 161 and 163. When servers 161 and 163 receive this command, the servers include write requests with time stamps earlier than the time transmitted with the commands 161 and 162 in the snapshot. Write requests with time stamps later than the time stamp of the take snapshot commands 160 and 162 are not included in the generated snapshot. In the example depicted in FIG. 13 the write requests 136 and 134 are included within the generated snapshot while the write request 138 is not included within the generated snapshot. Once the snapshot information is generated, the process 120 may proceed as the process 70 described above with reference to FIG. 12.

The methods of the present invention may be performed in either hardware, software, or any combination thereof, as those terms are currently known in the art. In particular, the present methods may be carried out by software, firmware, or microcode operating on a computer or computers of any type. Additionally, software embodying the present invention may comprise computer instructions in any form (e.g., source code, object code, interpreted code, etc.) stored in any computer-readable medium (e.g., ROM, RAM, magnetic media, punched tape or card, compact disc (CD) in any form, DVD, etc.). Furthermore, such software may also be in the form of a computer data signal embodied in a carrier wave, such as that found within the well-known Web pages transferred among devices connected to the Internet. Accordingly, the present invention is not limited to any particular platform, unless specifically stated otherwise in the present disclosure.

Moreover, the depicted systems and methods may be constructed from conventional hardware systems and specially developed hardware is not necessary. For example, in the depicted systems, the clients can be any suitable computer system such as a PC workstation, a handheld computing device, a wireless communication device, or any other such device equipped with a network client hardware and or software capable of accessing a network server and interacting with the server to exchange information. Optionally, the client and the server may rely on an unsecured communication path for accessing services on the remote server. To add security to such a communication path, the client and the server can employ a security system, such the Secured Socket Layer (SSL) security mechanism, which provides a trusted path between a client and a server. Alternatively, they may employ another conventional security system that has been developed to provide to the remote user a secured channel for transmitting data over a network.

Furthermore, networks employed in the systems herein disclosed may include conventional and unconventional computer to computer communications systems now known or engineered in the future, such as but not limited to the Internet.

Servers may be supported by a commercially available server platform such as a Sun Microsystems, Inc. Sparc ™ system running a version of the UNIX operating system and running a server program capable of connecting with, or exchanging data with, clients.

Those skilled in the art will know, or be able to ascertain using no more than routine experimentation, many equivalents to the embodiments and practices described herein. For example, the processing or I/O capabilities of the servers 161, 162, 163 may not be the same, and the allocation process 220 will take this into account when making resource

migration decisions. In addition, there may be several parameters that together constitute the measure of "load" in a system – network traffic, I/O request rates, as well as data access patterns (for example, whether the accesses are predominantly sequential or predominantly random). The allocation process 220 will take all these parameters into account as input to its migration decisions.

As discussed above, the invention disclosed herein can be realized as a software component operating on a conventional data processing system such as a UNIX workstation. In that embodiment, the short cut response mechanism can be implemented as a C language computer program, or a computer program written in any high level language including C++, C#, Pascal, FORTRAN, Java, or BASIC. Additionally, in an embodiment where microcontrollers or digital signal processors (DSPs) are employed, the short cut response mechanism can be realized as a computer program written in microcode or written in a high level language and compiled down to microcode that can be executed on the platform employed. The development of such code is known to those of skill in the art, and such techniques are set forth in Digital Signal Processing Applications with the TMS320 Family, Volumes I, II, and III, Texas Instruments (1990). Additionally, general techniques for high level programming are known, and set forth in, for example, Stephen G. Kochan, Programming in C, Hayden Publishing (1983).

While particular embodiments of the present invention have been shown and described, it will be apparent to those skilled in the art that changes and modifications may be made without departing from this invention in its broader aspect and, therefore, the appended claims are to encompass within their scope all such changes and modifications as fall within the true spirit of this invention.

CLAIMS

We claim:

1. An apparatus for resource migration, comprising a storage system having

a plurality of storage servers with a set of resources partitioned thereon, said storage servers having a load monitor process capable of communicating with other load monitor processes for generating a measure of loading on respective ones of the plurality of servers; and

a resource migration process for transferring a resource from one of said plurality of servers to another of said plurality of servers in response to said measure of loading.

2. The apparatus of Claim 1, wherein said servers are equivalent to each other.

3. The apparatus of Claim 1, wherein said resources are selected from the group consisting of data blocks, program files, multimedia files, applications, and database files.

4. The apparatus of Claim 1, wherein said measure of loading reflects both a storage system load and a server load.

5. The apparatus of Claim 1, wherein said storage system is a Storage Area Network.

6. The apparatus of Claim 1, wherein the load monitor includes a process to determine whether a server is servicing a disproportionate share of the client requests being handled by a server group.

7. The apparatus of Claim 1, wherein the resource migration process includes a block data migration process.

8. The apparatus of Claim 1, further including a routing table for tracking resources maintained on the system.

9. The apparatus of Claim 1, wherein a pointer to a resource is maintained during a an access operation to provide continuous data access.

10. The apparatus of Claim 1, wherein the load monitoring process monitors one or more of network traffic load, I/O request load, storage traffic pattern type.

11. The apparatus of Claim 1, wherein the resource migration process includes a further process to detect when a resource write request applies to a resource that is in the process of being moved from a first server to a second server, and apply such resource write request to both copies of the resource held at said first and said second server.

12. The apparatus of Claim 1, wherein the resource migration process divides the resource being moved into smaller subresources, such that each subresource is moved from a first server to a second server in turn, and recovery from failure requires only the recovery of the subresource being moved at the time of failure and subsequent subresources.

13. The apparatus of Claim 12, wherein the resource migration process includes a further process to detect when a resource write request applies to a subresource that is in the process of being moved from a first server to a second server, and apply such resource write request to both copies of the resource held at said first and said second server.

14. A process for moving resources across a storage system havinga plurality of storage servers with a set of resources partitioned thereon, comprising the steps of

monitoring a load on a server and communicating with other load monitor processes for generating a measure of loading on respective ones of the plurality of servers; and

transferring, as a function of the measured loads, a resource from one of said plurality of servers to another of said plurality of servers in response to said measure of loading.

15. The process of Claim 14, wherein said servers are equivalent to each other.

16. The process of Claim 14, measuring a load includes measuring a storage system load and a server load.

17. The process of Claim 14, including the further step determining whether a server is servicing a disproportionate share of the client requests being handled by a server group.

18. The process of Claim 14, wherein the resource migration process includes a block data migration process.

19. The process of Claim 14, further including maintaining a routing table for tracking resources maintained on the system.

20. The process of Claim 14, wherein the load monitoring process monitors one or more of network traffic load, I/O request load, storage traffic pattern type.

21. The process of Claim 14, further including maintaining a pointer to a resource is maintained during an access operation to provide continuous data access.

22. The process of Claim 14, further including detecting when a resource write request applies to a resource that is in the process of being moved from a first server to a second server, and applying such resource write request to both copies of the resource held at said first and said second server.

23. The process of Claim 14, further including dividing the resource being moved into smaller subresources, such that each subresource is moved from a first server to a second server in turn, and recovery from failure requires only the recovery of the subresource being moved at the time of failure and subsequent subresources.

24. The process of Claim 23, further including detecting when a resource write request applies to a subresource that is in the process of being moved from a first server to a second server, and apply such resource write request to both copies of the resource held at said first and said second server.

25. A system for managing requests from a plurality of clients for access to a set of resources, comprising a plurality of servers having the set of resources partitioned thereon, each server having

a load monitor process capable of communicating with other load monitor processes for generating a measure of system load, and

a client load on each of the plurality of servers.

26. A system according to claim 25, further comprising a client distribution process, responsive to the system load, and capable of repartitioning the set of client connections for distributing client load.

27. A system according to claim 25, further comprising a load distribution process for determining resource loads when moving clients among servers.

28. A system according to claim 25, further comprising a client allocation process for causing a client to communicate with a selected one of said plurality of servers.

29. A system according to claim 25, further comprising a client allocation process for distributing incoming client requests across said plurality of servers.

30. A system according to claim 26, wherein the client distribution process includes a round robin distribution process.

31. A system according to claim 26, wherein the client distribution process includes a client redirection process.

32. A system according to claim 26, wherein the client distribution process includes a disconnect process for dynamically disconnecting a client from a first server and reconnecting to a second server.

33. A system according to claim 25, further comprising an application program executing on at least one of the servers and being capable of transferring a client connection to a different server.

34. A system according to claim 25, further comprising an adaptive client distribution process for distributing clients across the plurality of servers as a function of dynamic variations is measured system load.

35. A system according to claim 25, further comprising a storage device for providing storage resources to the plurality of clients.

36. A system according to claim 25, further comprising a storage service process for providing at least one volume of storage partitioned across the plurality of servers.

37. A storage area network, comprising a plurality of servers each configured as a server of claim 25.

38. Systems for providing a partitioned storage service, comprising

at least two servers,

a storage volume partitioned across the at least two servers, and

at least two snapshot processes operating on respective ones of the at least two servers and capable of coordinating with other snapshot processes for generating state information representative of the state of the partitioned storage volume.

39. Systems according to claim 38, wherein the snapshot process includes a coordinating process for generating commands for coordinating at least one other snapshot process to generate state information representative of the state of the partitioned storage volume.

40. Systems according to claim 39, wherein the coordinating process includes a time-stamp process for time stamping a command to generate a snapshot process.

41. Systems according to claim 38, wherein the snapshot process includes a request control process for processing requests received by the respective server.

42. Systems according to claim 41, wherein the request control process includes a suspend process for suspending processing of requests by the respective server.

43. Systems according to claim 41, wherein the request control process includes a time-stamp process for time stamping requests received by the respective server.

44. Systems according to claim 38, wherein the snapshot process includes process for analyzing suspended requests to determine requests received after a selected time.

45. Systems according to claim 38, further including an archive process for employing the state information to create a copy of the storage volume.

46. Systems according to claim 38, further comprising a plurality of storage volumes partitioned across the at least two servers.

47. A process for providing a partitioned storage service, comprising the steps of

providing at least two servers and a storage volume partitioned across the at least two servers, and

operating at least two snapshot processes on respective ones of the at least two servers and capable of coordinating with other snapshot processes for generating state information representative of the state of the partitioned storage volume.

48. A process according to claim 47, including coordinating at least one other snapshot process to generate state information representative of the state of the partitioned storage volume.

49. A process according to claim 48, wherein coordinating includes time-stamping a command to generate a snapshot process.

50. A process according to claim 47, wherein operating a snapshot process includes operating a request control process for processing requests received by the respective server.

51. A process according to claim 50, wherein the request control process includes a suspend process for suspending processing of requests by the respective server.

52. A process according to claim 50, wherein the request control process time stamps requests received by the respective server.

53. A process according to claim 50, further including analyzing suspended requests to determine requests received after a selected time.

54. A process for generating a snapshot of a storage volume distributed across at least two servers, comprising

executing snapshot processes on respective ones of the at least two servers,

providing an administration command to a first one of the snapshot processes directing the snapshot processes to generate state information representative of the state of the partitioned storage volume,

having the first snapshot process hold pending requests and direct at least a second snapshot process to hold pending client requests,

having the second snapshot process to indicate that requests have been held, and

having the first snapshot process generate state information representative of the state of a storage partition maintained on its respective server and generate a snapshot command for the second server to generate information representative of the state of a storage partition maintained on its respective server.

55. A process according to claim 54, wherein the administration command includes a prepare command to a second server supporting the data volume for which a snapshot is being created.

56. A process according to claim 54, further comprising processing the state information to generate an archive copy of the storage volume.

57. A process according to claim 54, further comprising having the first and second snapshot processes release pending requests after generating the state information.

58. A storage area network, comprising

a data network having at least two servers,

a storage volume partitioned across the at least two servers, and

at least two snapshot processes operating on respective ones of the at least two servers and capable of coordinating with other snapshot processes for generating state information representative of the state of the partitioned storage volume.

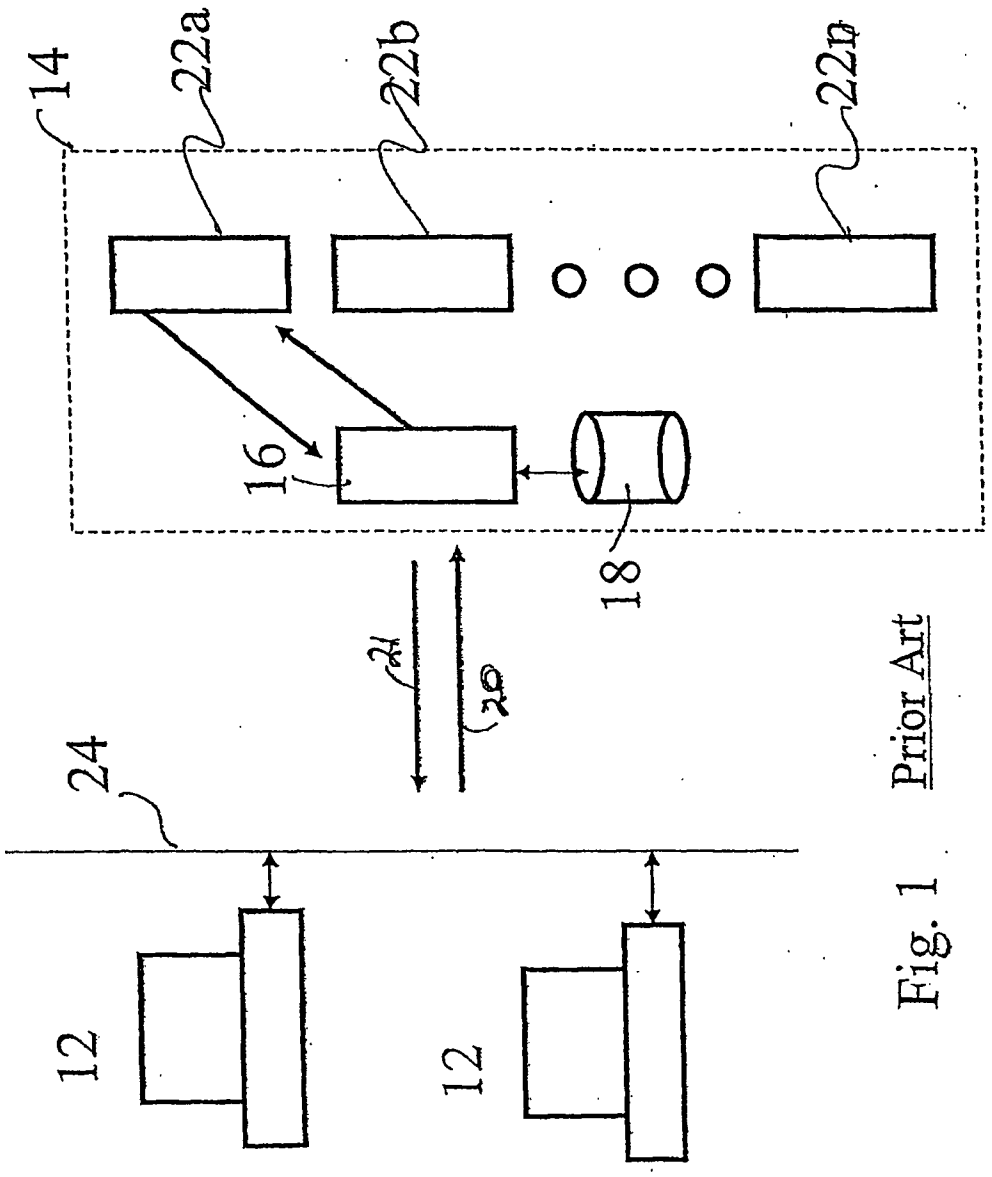


Fig. 1 Prior Art

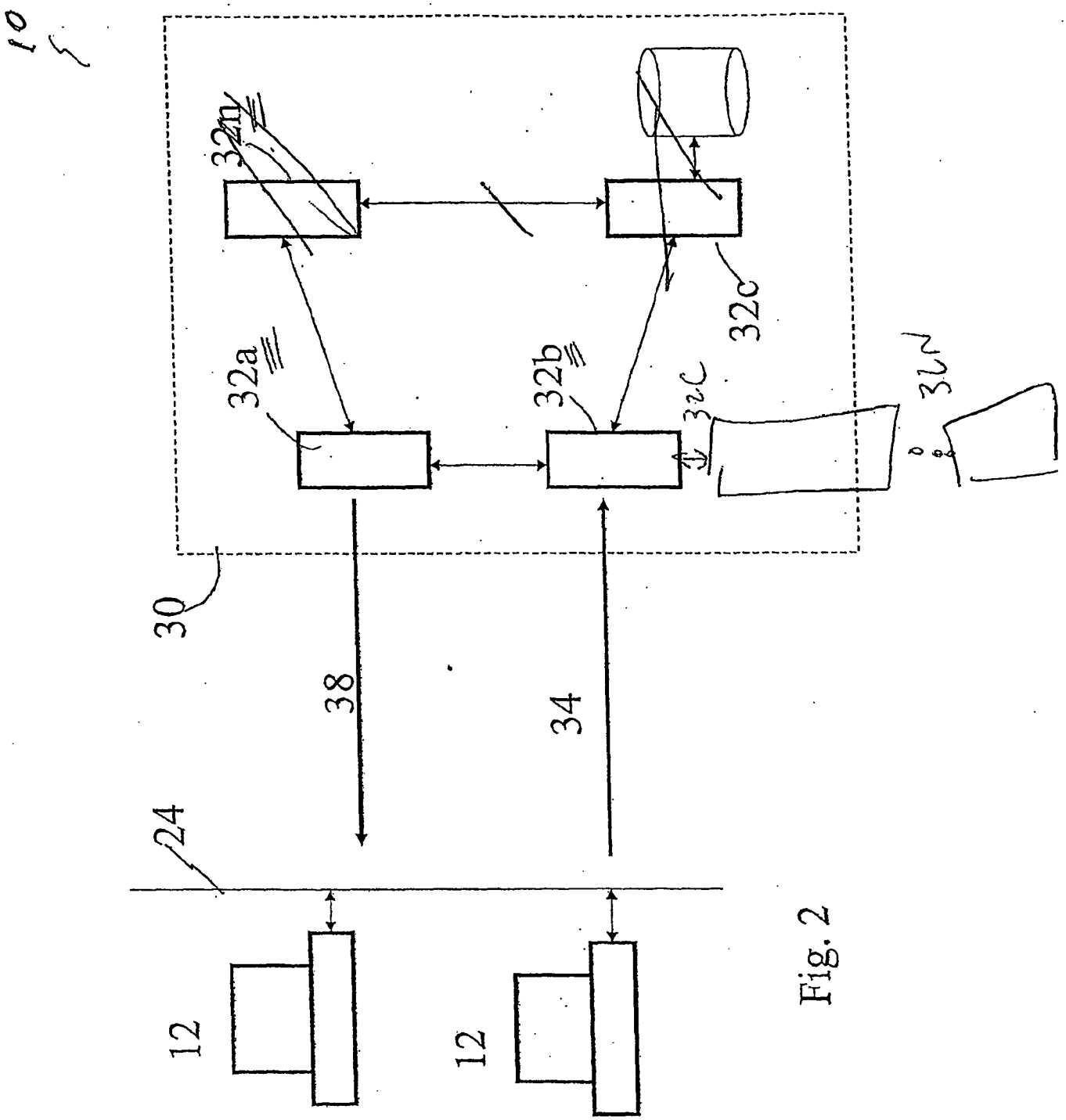


Fig. 2

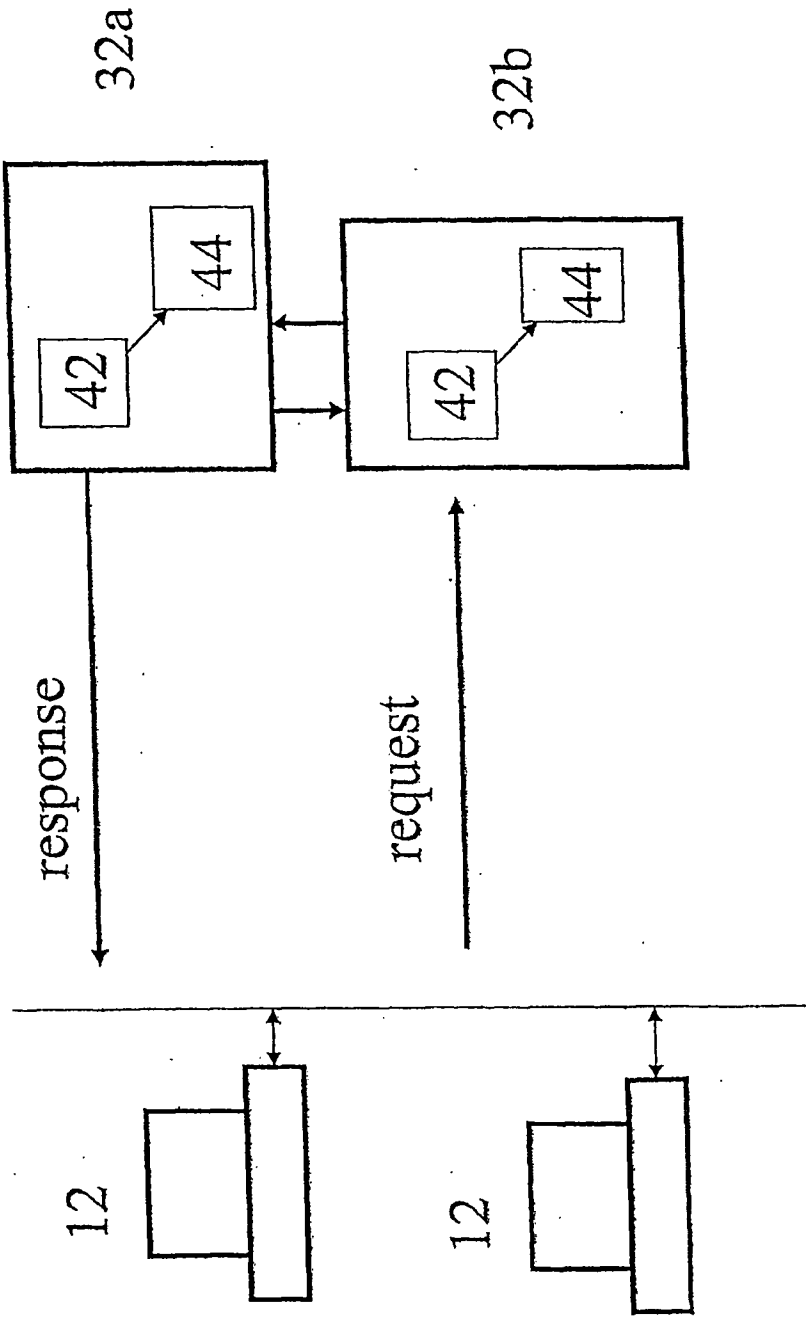


Fig. 3

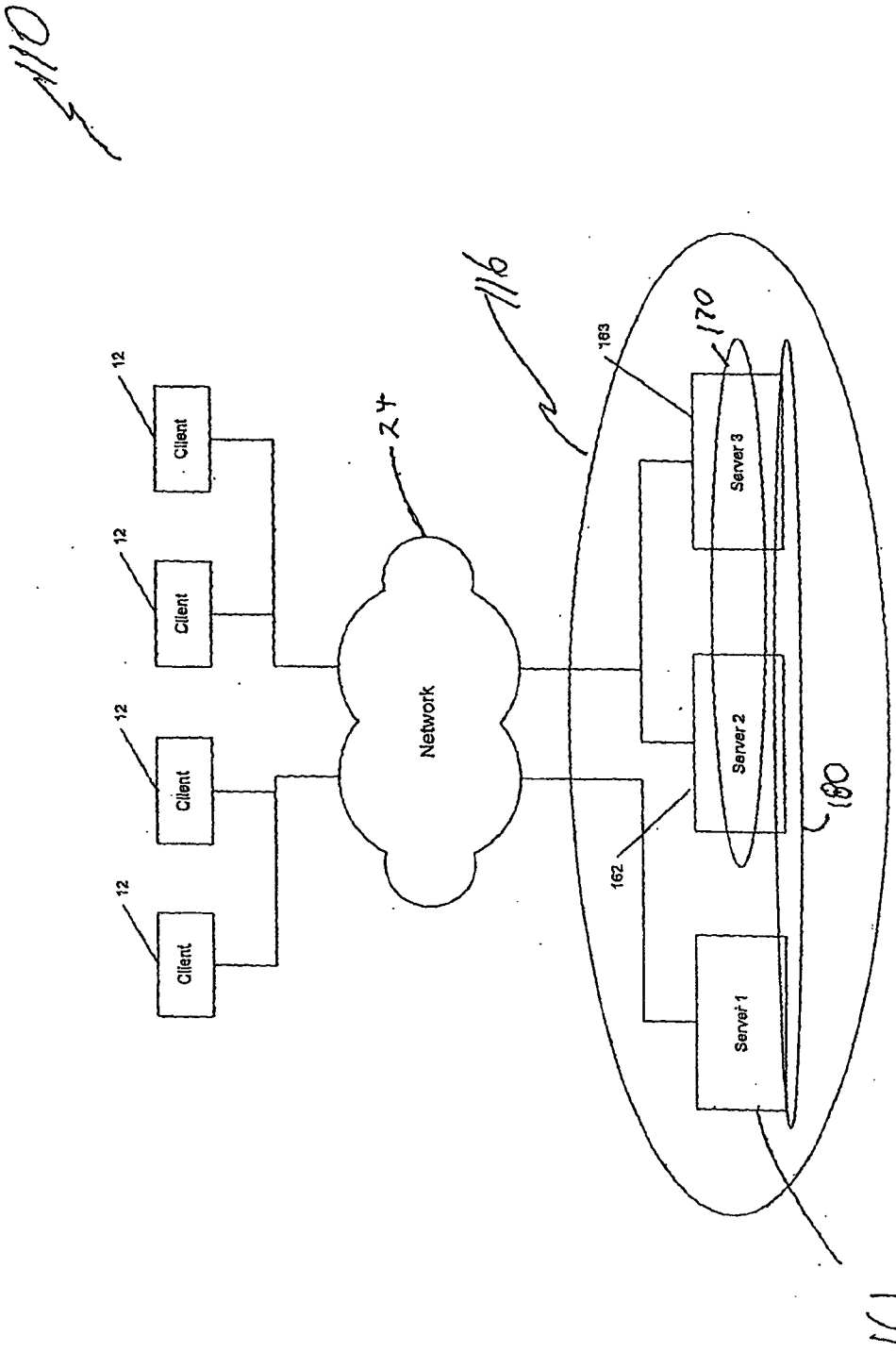

110
116
170
183
24
12
Client
Client
Client
Client
Network
Server 3
Server 2
Server 1
180
162
161

FIG. 4

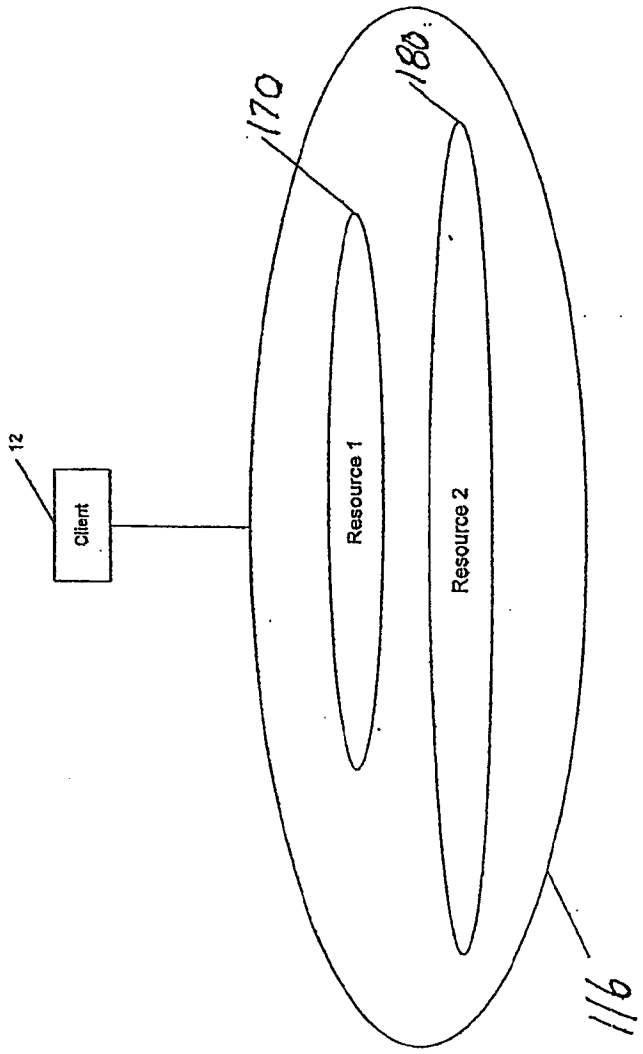


FIG. 5

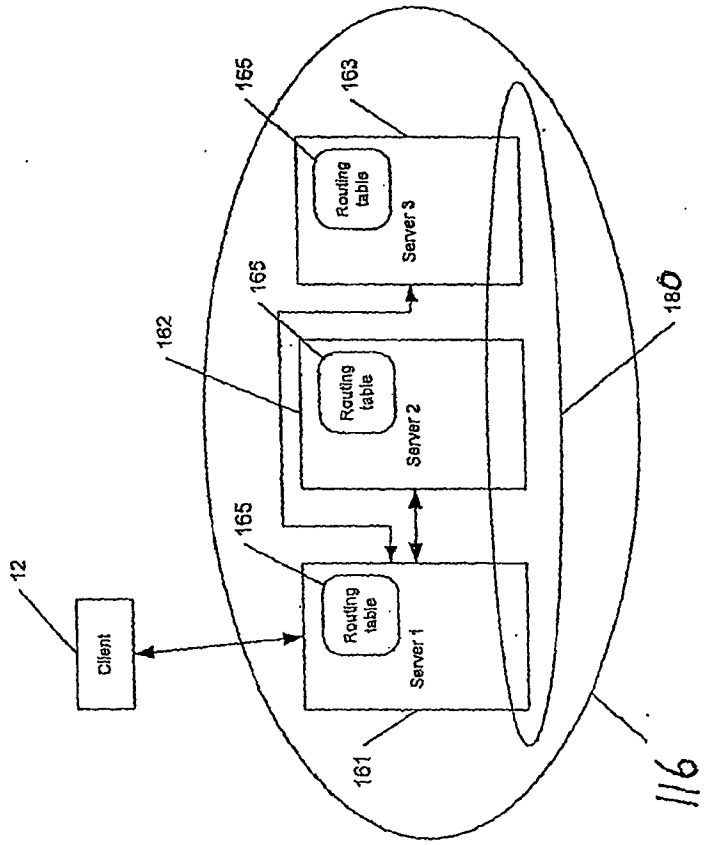

12
Client
161
Server 1
Routing table
165
162
Server 2
Routing table
165
163
Server 3
Routing table
165
180
116

FIG. 6

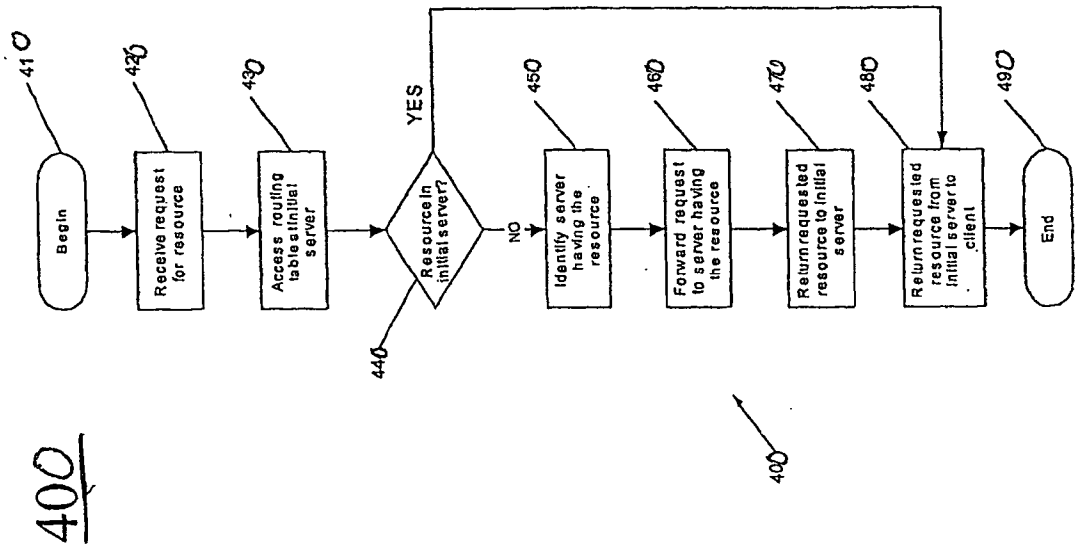

410
Begin
420
Receive request for resource
430
Access routing table at initial server
440
Resource in initial server?
YES
NO
450
Identify server having the resource
460
Forward request to server having the resource
470
Return requested resource to initial server
480
Return requested resource from initial server to client
490
End
400

FIG. 7

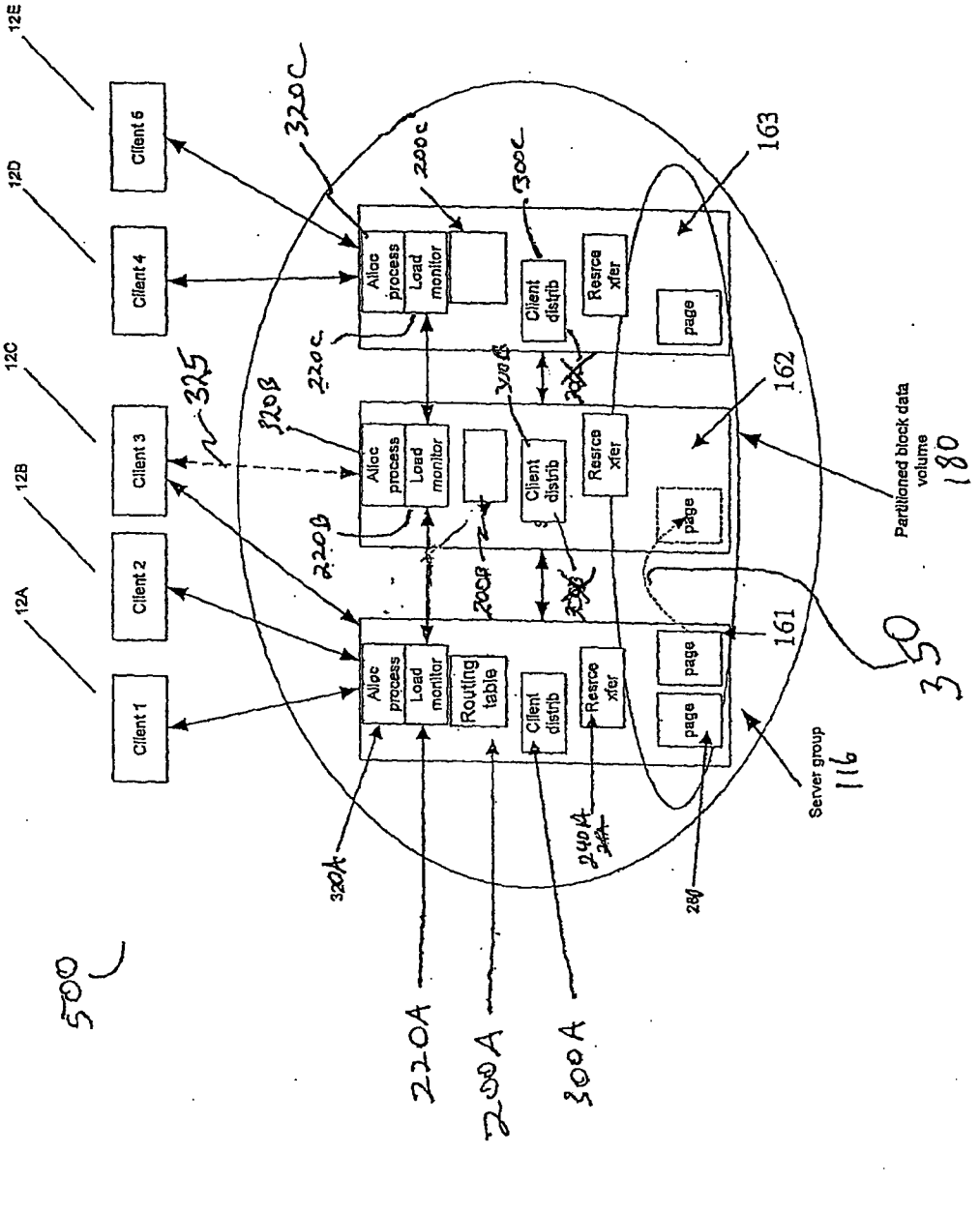

Client 1
Client 2
Client 3
Client 4
Client 6
12A
12B
12C
12D
12E
Alloc process
Load monitor
Routing table
Client distrib
Resrce xfer
page
Server group
116
Partitioned block data volume
180
161
162
163
350
500
220A
200A
300A
220B
200B
220C
200C
325

FIG. 8

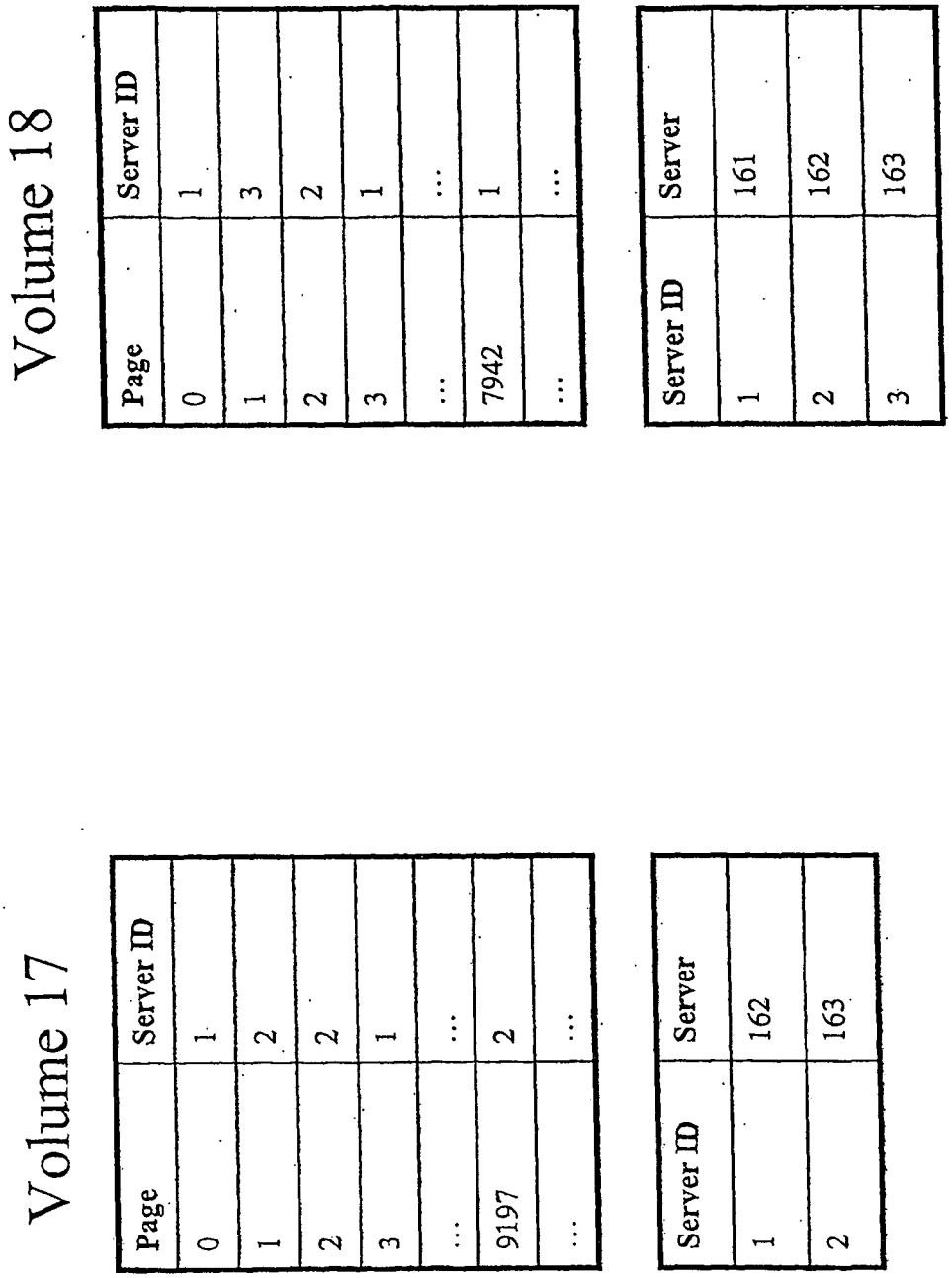

Volume 17
Page
Server ID
0
1
1
2
2
2
3
1
...
...
9197
2
...
...
Server ID
Server
1
162
2
163
Volume 18
Page
Server ID
0
1
1
3
2
2
3
1
...
...
7942
1
...
...
Server ID
Server
1
161
2
162
3
163

FIG. 9

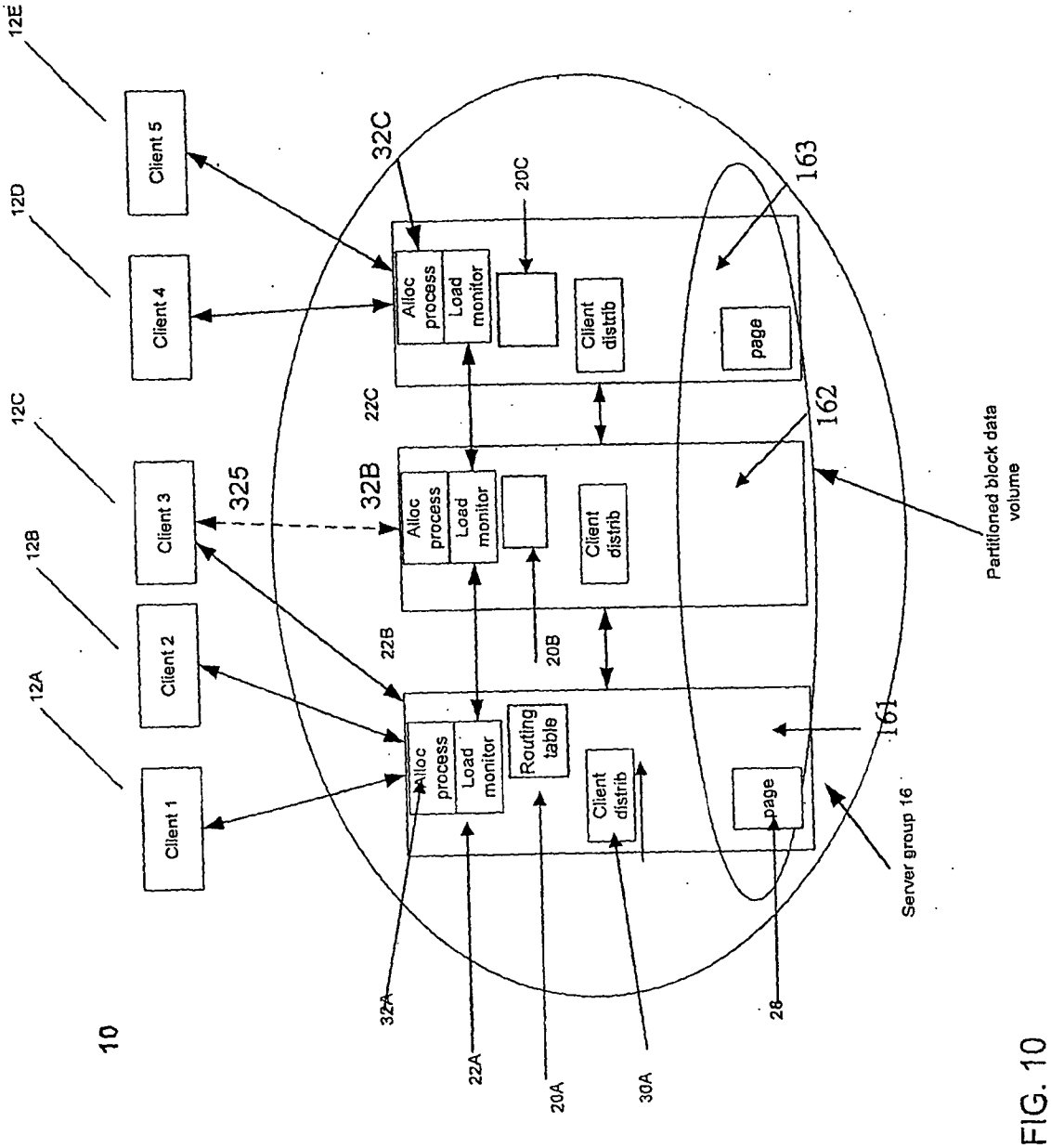

12E
Client 5
12D
Client 4
12C
Client 3
12B
Client 2
12A
Client 1
10
325
32C
20C
163
Alloc process
Load monitor
Client distrib
page
22C
32B
162
Partitioned block data volume
20B
22B
Routing table
161
Server group 16
32A
22A
20A
30A
28

FIG. 10

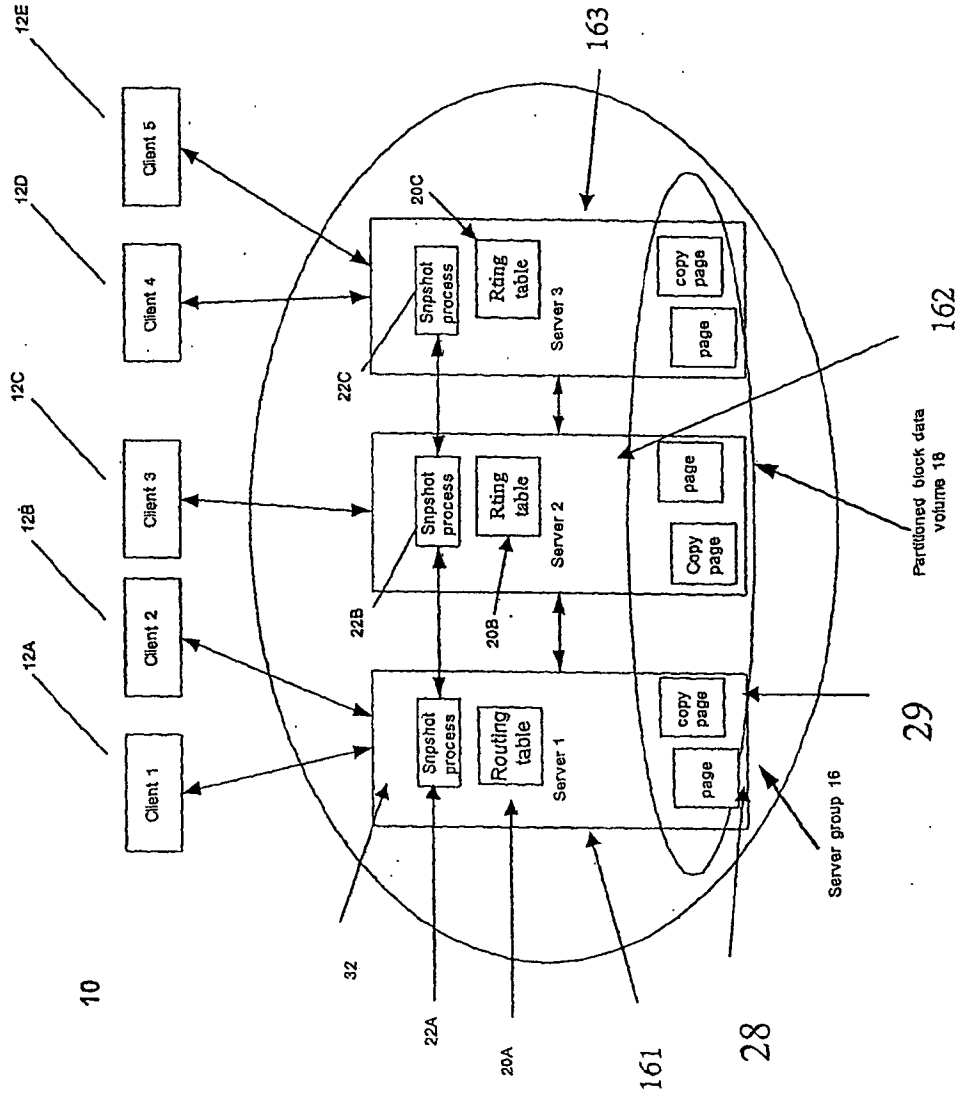
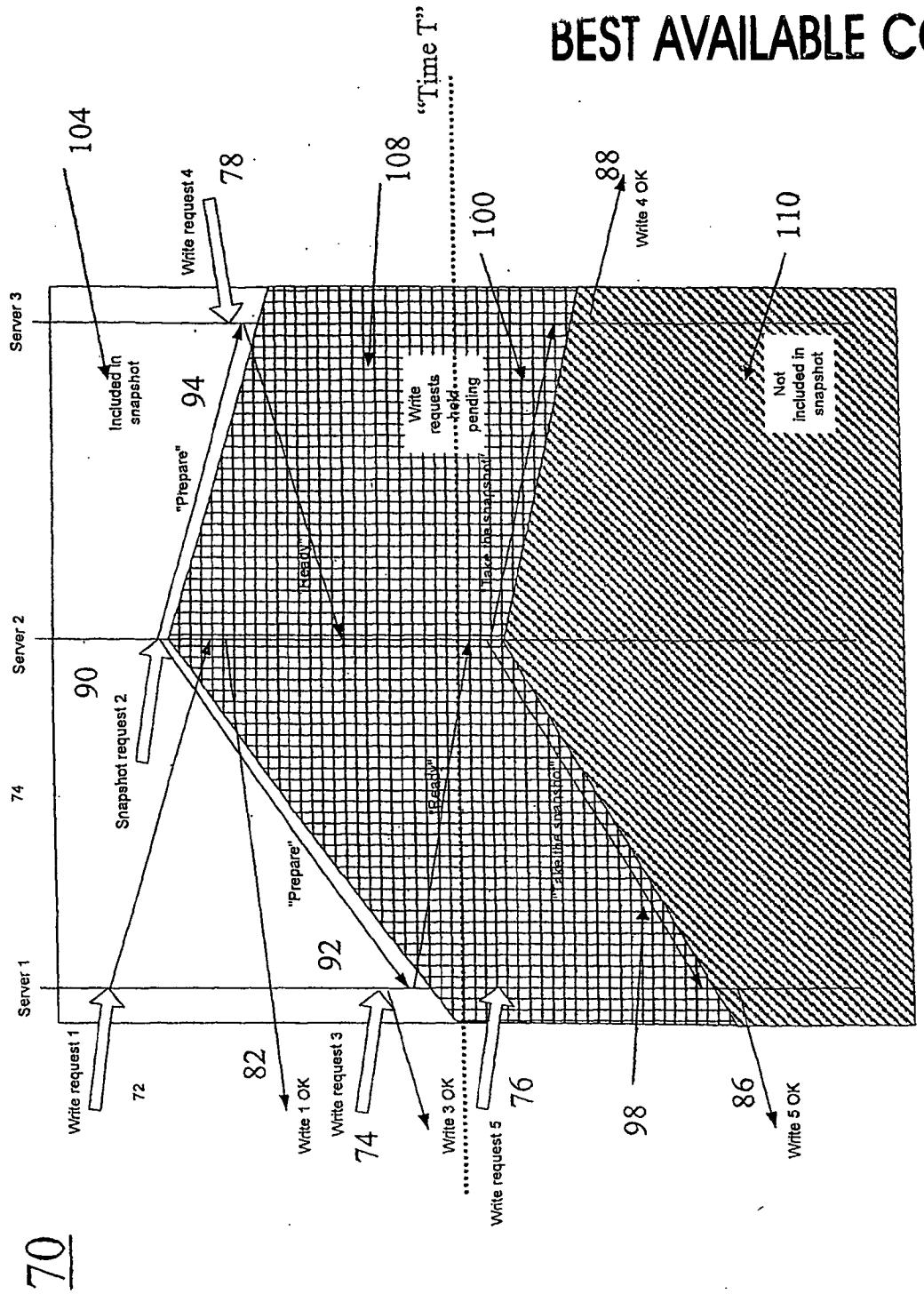

12E
12D
12C
12B
12A
Client 5
Client 4
Client 3
Client 2
Client 1
163
20C
Snpshot process
Ring table
Server 3
copy page
page
22C
162
Partitioned block data volume 18
Snpshot process
Ring table
Server 2
page
Copy page
22B
20B
Snpshot process
Routing table
Server 1
copy page
page
Server group 16
29
32
22A
20A
161
28
10

FIG. 11



70

FIG. 12

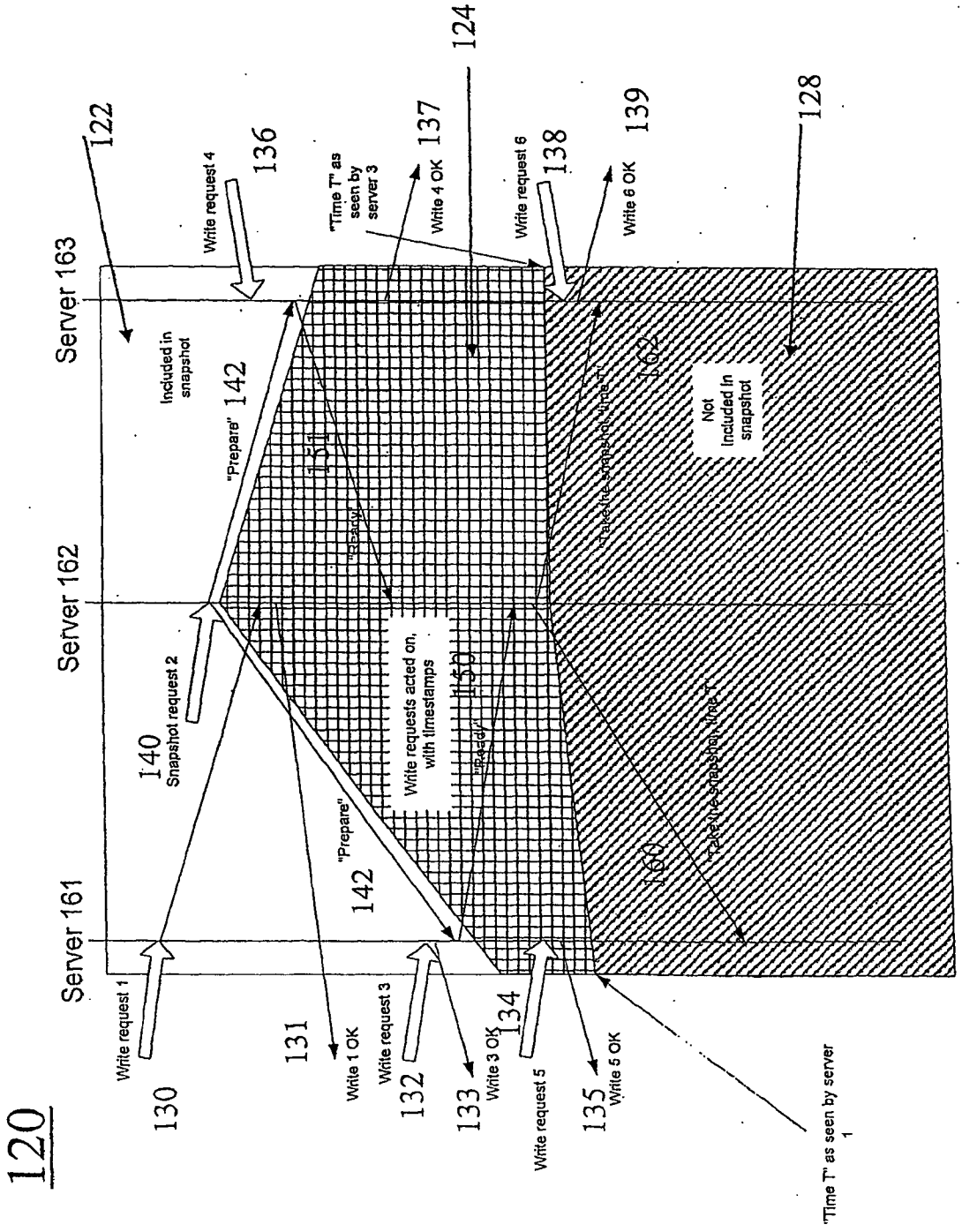

120
Server 161
Server 162
Server 163
122
124
128
130
131
132
133
134
135
136
137
138
139
140
142
150
151
160
162
Write request 1
Write 1 OK
Write request 3
Write 3 OK
Write request 5
Write 5 OK
Write request 4
"Time T" as seen by server 3
Write 4 OK
Write request 6
Write 6 OK
Snapshot request 2
"Prepare"
"Prepare"
Included in snapshot
Not Included in snapshot
Write requests acted on, with timestamps
Ready
Ready
Take the snapshot, time T
Take the snapshot, time T
"Time T" as seen by server 1

FIG. 13

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
5 August 2004 (05.08.2004)

PCT

(10) International Publication Number
WO 2004/066278 A3

(51) International Patent Classification[7]: G06F 17/30, 9/50

(21) International Application Number:
PCT/US2004/001632

(22) International Filing Date: 21 January 2004 (21.01.2004)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/441,810 21 January 2003 (21.01.2003) US
10/761,884 20 January 2004 (20.01.2004) US

(71) Applicant *(for all designated States except US)*: **EQUALLOGIC, INC.** [US/US]; 9 Townsend West, Nashua, NH 03063 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **KONING, Paul, G.** [US/US]; 408 Joe English Road, New Boston, NH 03070 (US). **HAYDEN, Peter, C.** [US/US]; 17 Purgatory Road, Mount Vernon, NH 03057 (US). **LONG, Paula** [US/US]; 25 Winchester Drive, Hollis, NH 03049 (US). **SUMAN, Daniel, E.** [US/US]; 11 Grizzley Bear Circle, Suite 201, Westford, MA 01886 (US). **LEE, Hsin, H.** [US/US]; 9 Townsend West, Nashua, NH 03063 (US).

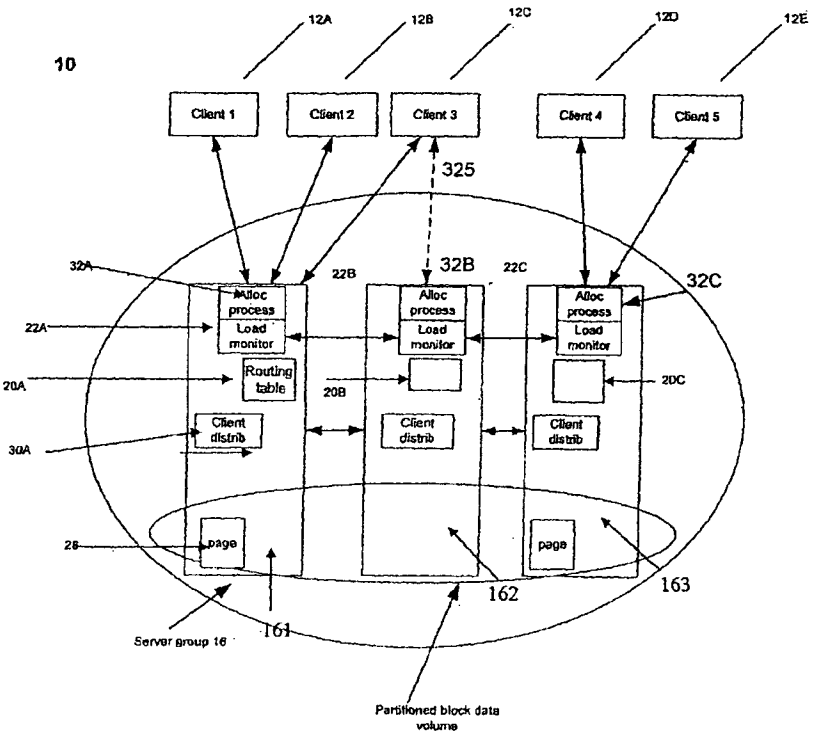
(74) Agents: **STUTIUS, Wolfgang, E.** et al.; Ropes & Gray LLP, Patent Group, One International Place, Boston, MA 02110-2624 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR,

[Continued on next page]

(54) Title: SYSTEMS FOR MANAGING DATA STORAGE



(57) Abstract: Systems for managing data storage are described. The systems manage responses to requests from a plurality of clients for access to a set of resources, and more efficiently responds to client load changes in storage area network (SAN) by migrating data blocks while providing continuous data access. The systems include a plurality of optionally equivalent servers wherein the set of resources is partitioned across these servers. Each (equivalent) server has a load monitor process that can communicate with the other load monitor processes for generating a measure of the client load on the server system and the client load on each of the respective servers. The system further comprises a resource distribution process that redistribute the client load by repartitioning the set of resources in response to the measured system load. In addition, each server may include a routing table that includes a reference for each resource that is maintained on the partitioned resource server. Requests from a client are processed as a function of the routing table to route the request to the individual server that maintains or has control over the resource of interest. For archiving purposes, a snapshot process may operate on a server, optionally in cooperation with other snapshot processes for generating state information representative of the state of the partitioned storage volume.

WO 2004/066278 A3

GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

(88) Date of publication of the international search report:
2 March 2006

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US2004/001632

A. CLASSIFICATION OF SUBJECT MATTER
G06F17/30 G06F9/50

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, INSPEC, IBM-TDB, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	SCHEUERMANN P ET AL: "DATA PARTITIONING AND LOAD BALANCING IN PARALLEL DISK SYSTEMS" TECHNICAL REPORT A/02/96 UNIVERSITY OF SAARLAND, April 1996 (1996-04), pages 1-48, XP002329842 Retrieved from the Internet: URL:ftp://cs.uni-sb.de/pub/techreports/FB14/fb14-96-02.ps.gz>	1-10, 14-21
Y	page 3, line 1 - page 7, line 21	25-37
A	page 16, line 23 - page 223, line 22	12,23

-/--

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
10 June 2005	23. 12. 2005

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Kielhöfer, P

Form PCT/ISA/210 (second sheet) (January 2004)

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	WO 99/53415 A (HEWLETT-PACKARD COMPANY; WOLFF, JAMES, J) 21 October 1999 (1999-10-21)	25-37
A	abstract page 3, line 26 - page 7, line 4 page 10, line 22 - page 17, line 27 page 23, line 5 - page 24, line 7 page 25, line 22 - page 26, line 2 page 26, line 15 - page 27, line 10 page 29, line 1 - page 30, line 28 page 32, line 1 - page 39, line 11 page 43, line 1 - line 21 page 45, line 8 - page 51, line 6 page 52, line 20 - page 53, line 10 page 56, line 20 - page 63, line 9 page 70, line 13 - page 72, line 28 figure 10G	1-10, 14-21

X	WEI LIU ET AL: "Design of an I/O balancing file system on web server clusters" PARALLEL PROCESSING, 2000. PROCEEDINGS. 2000 INTERNATIONAL WORKSHOPS ON 21-24 AUGUST 2000, PISCATAWAY, NJ, USA,IEEE, 21 August 2000 (2000-08-21), pages 119-125, XP010511941 ISBN: 0-7695-0771-9	1-6,9, 10, 14-17, 20,21, 25,35-37
A	the whole document	7,8,18, 19,28

A	ANDERSON T E ET AL: "SERVERLESS NETWORK FILE SYSTEMS" ACM TRANSACTIONS ON COMPUTER SYSTEMS, ASSOCIATION FOR COMPUTING MACHINERY. NEW YORK, US, vol. 14, no. 1, 1 February 1996 (1996-02-01), pages 41-79, XP000584662 ISSN: 0734-2071 page 48, line 24 - page 56, last line page 62, line 1 - page 65, line 23	1-37

	-/--	

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	HAC A ET AL INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS: "Dynamic load balancing in a distributed system using a decentralized algorithm" INTERNATIONAL CONFERENCE ON DISTRIBUTED COMPUTING SYSTEMS. WEST BERLIN, SEPT. 21 - 25, 1987, PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON DISTRIBUTED COMPUTING SYSTEMS, WASHINGTON, IEEE COMP. SOC. PRESS, US, vol. CONF. 7, 21 September 1987 (1987-09-21), pages 170-177, XP002105823 the whole document ----	1-6,10, 14-17, 20,25, 35-37

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2004/001632

Box II Observations where certain claims were found unsearchable (Continuation of item 2 of first sheet)

This International Search Report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:

2. ☐ Claims Nos.:
because they relate to parts of the International Application that do not comply with the prescribed requirements to such an extent that no meaningful International Search can be carried out, specifically:

3. ☐ Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box III Observations where unity of invention is lacking (Continuation of item 3 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

see additional sheet

1. ☐ As all required additional search fees were timely paid by the applicant, this International Search Report covers all searchable claims.

2. ☐ As all searchable claims could be searched without effort justifying an additional fee, this Authority did not invite payment of any additional fee.

3. ☐ As only some of the required additional search fees were timely paid by the applicant, this International Search Report covers only those claims for which fees were paid, specifically claims Nos.:

4. ☒ No required additional search fees were timely paid by the applicant. Consequently, this International Search Report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

1-37

Remark on Protest

☐ The additional search fees were accompanied by the applicant's protest.

☐ No protest accompanied the payment of additional search fees.

FURTHER INFORMATION CONTINUED FROM PCT/ISA/ 210

This International Searching Authority found multiple (groups of) inventions in this international application, as follows:

1. claims: 1-37

Apparatus, method and system for load balancing a partitioned distributed storage system

---

2. claims: 38-58

Systems and methods for generating a snapshot of partitioned storage volumes

---

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9953415 A	21-10-1999	AU 3861399 A	01-11-1999